Administración de red

Remo Suppi Boldrito

P07/M2103/02285



Índice

Introducción

El sistema operativo UNIX (GNU/Linux) se toma como ejemplo de una arquitectura de comunicaciones estándar. Desde el mítico UUCP (servicio de copia entre sistemas operativos UNIX) hasta las redes actuales, UNIX siempre ha mostrado su versatilidad en aspectos relacionados con la comunicación y el intercambio de información. Con la introducción de redes de ordenadores (área local LAN, área amplia WAN o las más actuales área metropolitana MAN) ofreciendo enlaces multipunto a diferentes velocidades (56kbits/seg hasta 1Gbit/seg), han ido surgiendo nuevos servicios basados en protocolos más rápidos, portables entre diferentes ordenadores y mejor adaptados, como el TCP/IP (*transport control program / Internet protocol*). [Com01, Mal96, Cis00, Gar98, KD00]

1. Introducción a TCP/IP (TCP/IP suite)

El protocolo TCP/IP sintetiza un ejemplo de estandarización y una voluntad de comunicación a nivel global.

El protocolo TCP/IP es en realidad un conjunto de protocolos básicos que se han ido agregando a principal para satisfacer las diferentes necesidades en la comunicación ordenador-ordenador como son TCP, UDP, IP, ICMP, ARP. [Mal96]

La utilización más frecuente de TCP/IP para el usuario en la actualidad es la conexión remota a otros ordenadores (telnet, SSH Secure Shell), la utilización de ficheros remotos (*network file system* NFS) o su transferencia (*file transfer protocol* FTP, *hipertext markup protocol*, HTTP).

Nota

Utilización típica de TCP/IP remote login: `telnet localhost` Debian GNU/ Linux 4.0 login:

1.1. Servicios sobre TCP/IP

Los servicios TCP/IP tradicionales más importantes son [Gar98]:

- **Transferencia de archivos**: el *file transfer protocol (*FTP) permite a un usuario de un ordenador obtener/enviar archivos de un ordenador hacia otro ordenador. Para ello, el usuario deberá tener una cuenta en el ordenador remoto e identificarse a través de su nombre (*login*) y una palabra clave (*password*) o a ordenadores donde existe un repositorio de información (software, documentación...), y el usuario se conectará como anónimo (*anonymous*) para transferir (leer) estos archivos a su ordenador. Esto no es lo mismo que los más recientes sistemas de archivos de red, NFS, Network File System, (o protocolos netbios sobre tcp/ip, "invento" totalmente inseguro sobre Windows y que es mejor reemplazar por una versión más antigua pero más segura del mismo concepto llamado netbeui) que permiten virtualizar el sistema de archivos de una máquina para que pueda ser accedido en forma interactiva sobre otro ordenador.

- **Conexión** (*login*) remota: el protocolo de terminal de red (telnet) permite a un usuario conectarse a un ordenador remotamente. El ordenador local se utiliza como terminal del ordenador remoto y todo es ejecutado sobre éste permaneciendo el ordenador local invisible desde el punto de vista de la sesión. Este servicio en la actualidad se ha reemplazado por el SHH (*secure shell*) por razones de seguridad. En una conexión remota mediante telnet, los mensajes circulan tal cual (texto plano), o sea, si alguien "observa" los mensajes en la red, equivaldrá a mirar la pantalla del usuario. SSH

codifica la información (que significa un coste añadido a la comunicación) que hace que los paquetes en la red sean ilegibles a un nodo extraño.

- **eMail**: este servicio permite enviar mensajes a los usuarios de otros ordenadores. Este modo de comunicación se ha transformado en un elemento vital en la vida de los usuarios y permite que los *e-mails* (correos electrónicos) sean enviados a un servidor central para que después puedan ser recuperados por medio de programas específicos (clientes) o leídos a través de una conexión web.

El avance de la tecnología y el bajo coste de los ordenadores ha permitido que determinados servicios se hayan especializado en ello y se ofrecen configurados sobre determinados ordenadores trabajando en un modelo cliente-servidor. Un servidor es un sistema que ofrece un servicio específico para el resto de la red. Un cliente es otro ordenador que utiliza este servicio. Todos estos servicios generalmente son ofrecidos dentro de TCP/IP:

- **Sistemas de archivos en red** (*network file systems*): permite a un sistema acceder a los archivos sobre un sistema remoto en una forma más integrada que FTP. Los dispositivos de almacenamiento (o parte de ellos) son exportados hacia el sistema que desea acceder y éste los puede "ver" como si fueran dispositivos locales. Este protocolo permite a quien exporta poner las reglas y las formas de acceso, lo que (bien configurado) hace independiente el lugar donde se encuentra la información físicamente del sitio donde se "ve" la información.

- **Impresión remota**: permite acceder a impresoras conectadas a otros ordenadores.

- **Ejecución remota**: permite que un usuario ejecute un programa sobre otro ordenador. Hay diferentes maneras de realizar esta ejecución: o bien a través de un comando (rsh, ssh, rexec) o a través de sistemas con RPC (*remote procedure call*), que permiten a un programa en un ordenador local ejecutar una función de un programa sobre otro ordenador. Los mecanismos RPC han sido objeto de estudio y existen diversas implementaciones, pero las más comunes son Xerox's Courier y Sun's RPC (esta última adoptada por la mayoría de los UNIX).

- **Servidores de nombre** (*name servers*): en grandes instalaciones existen un conjunto de datos que necesitan ser centralizados para mejorar su utilización, por ejemplo, nombre de usuarios, palabras claves, direcciones de red, etc. Tod ello facilita que un usuario disponga de una cuenta para todas las máquinas de una organización. Por ejemplo, Sun's Yellow Pages (NIS en las versiones actuales de *Sun*) está diseñado para manejar todo este tipo de datos y se encuentra disponible para la mayoría de UNIX. El DNS (*domain name system*) es otro servicio de nombres pero que guarda una relación en-

tre el nombre de la máquina y la identificación lógica de esta máquina (dirección IP).

- **Servidores de terminal** (*terminal servers*): conecta terminales a un servidor que ejecuta telnet para conectarse al ordenador central. Este tipo de instalaciones permite básicamente reducir costes y mejorar las conexiones al ordenador central (en determinados casos).

- **Servidores de terminales gráficas** (*network-oriented window systems*): permiten que un ordenador pueda visualizar información gráfica sobre un *display* que está conectado a otro ordenador. El más común de estos sistemas es X Window.

1.2. ¿Qué es TCP/IP?

TCP/IP son en realidad dos protocolos de comunicación entre ordenadores independientes uno del otro.

Por un lado, TCP (*transmission control protocol*), define las reglas de comunicación para que un ordenador (*host*) pueda 'hablar' con otro (si se toma como referencia el modelo de comunicaciones OSI/ISO describe la capa 4, ver tabla siguiente).

TCP es orientado a conexión, es decir, equivalente a un teléfono, y la comunicación se trata como un flujo de datos (*stream*).

Por otro lado, IP (*Internet protocol*), define el protocolo que permite identificar las redes y establecer los caminos entre los diferentes ordenadores.

Es decir, encamina los datos entre dos ordenadores a través de las redes. Corresponde a la capa 3 del modelo OSI/ISO y es un protocolo sin conexión (ver tabla siguiente). [Com01, Rid00, Dra99]

Una alternativa al TCP la conforma el protocolo UDP (*user datagram protocol*), el cual trata los datos como un mensaje (datagrama) y envía paquetes. Es un protocolo sin conexión (el ordenador destino no debe necesariamente estar escuchando cuando un ordenador establece comunicación con él) y tiene la ventaja de que ejerce una menor sobrecarga a la red que las conexiones de TCP, pero la comunicación no es fiable (los paquetes pueden no llegar o llegar duplicados).

Existe otro protocolo alternativo llamado ICMP (*Internet control message protocol*). ICMP se utiliza para mensajes de error o control. Por ejemplo, si uno in-

tenta conectarse a un equipo (*host*), el ordenador local puede recibir un mensaje ICMP indicando "*host unreachable*". ICMP también puede ser utilizado para extraer información sobre una red. ICMP es similar a UDP, ya que maneja mensajes (datagramas), pero es más simple que UDP, ya que no posee identificación de puertos (los puertos son buzones donde se depositan los paquetes de datos y desde donde las aplicaciones servidoras leen dichos paquetes) en el encabezamiento del mensaje.

En el modelo de comunicaciones de la OSI/ISO (OSI, *open systems interconnection reference model,* ISO, *international standards organization*), es un modelo teórico adoptado por muchas redes. Existen siete capas de comunicación donde cada una tiene una interfaz para comunicarse con la anterior y la posterior:

Nivel	Nombre	Utilización
7	Aplicación	SMTP, *simple mail transfer protocol*, el servicio propiamente dicho
6	Presentación	Telnet, FTP implementa el protocolo del servicio
5	Sesión	Generalmente no se utiliza
4	Transporte	TCP, UDP transformación de acuerdo al protocolo de comunicación
3	Red	IP permite encaminar el paquete (*routing*)
2	Link	Controladores (*drivers*) transformación de acuerdo al protocolo físico
1	Físico	Ethernet, ADSL... envía del paquete físicamente

En resumen, TCP/IP es una familia de protocolos (que incluyen IP, TCP, UDP), que proveen un conjunto de funciones a bajo nivel utilizadas por la mayoría de las aplicaciones. [KD00, Dra99].

Algunos de los protocolos que utilizan los servicios mencionados han sido diseñados por Berkeley, Sun u otras organizaciones. Ellos no forman oficialmente parte de *Internet protocol suite* (IPS). Sin embargo, son implementados utilizando TCP/IP y por lo tanto considerados como parte formal de IPS. Una descripción de los protocolos disponibles en Internet puede consultarse en la RFC 1011 (ver referencias sobre RFC [IET]), que lista todos los protocolos disponibles. Existe actualmente una nueva versión del protocolo IPv6, también llamado IPng (IP *next generation*) que reemplaza al IPv4. Este protocolo mejora notablemente el anterior en temas tales como mayor número de nodos, control de tráfico, seguridad o mejoras en aspectos de *routing*.

1.3. Dispositivos físicos (hardware) de red

Desde el punto de vista físico (capa 1 del modelo OSI), el hardware más utilizado para LAN es conocido como Ethernet (o FastEthernet o GigaEthernet). Sus ventajas son su bajo coste, velocidades aceptables (10, 100, o 1.000 megabits por segundo) y facilidad en su instalación.

Existen tres modos de conexión en función del tipo de cable de interconexión: grueso (*thick*), fino (*thin*), y par trenzado (*twisted par*).

Las dos primeras están obsoletas (utilizan cable coaxial), mientras que la última se realiza a través de cables (pares) trenzados y conectores similares a los telefónicos (se conocen como RJ45). La conexión par trenzado es conocida como 10baseT o 100baseT (según la velocidad) y utiliza repetidores llamados *hubs* como puntos de interconexión. La tecnología Ethernet utiliza elementos intermedios de comunicación (*hubs*, *switchs*, *routers*) para configurar múltiples segmentos de red y dividir el tráfico para mejorar las prestaciones de transferencia de información. Normalmente, en las grandes instituciones estas LAN Ethernet están interconectadas a través de fibra óptica utilizando tecnología FDDI (*fiber distributed data interface*) que es mucho más cara y compleja de instalar, pero se pueden obtener velocidades de transmisión equivalentes a Ethernet y no tienen la limitación de la distancia de ésta (FDDI admite distancias de hasta 200 km). Su coste se justifica para enlaces entre edificios o entre segmentos de red muy congestionados.[Rid00, KD00].

Existe además otro tipo de hardware menos común, pero no menos interesante, como es ATM (*asynchronous transfer mode*). Este hardware permite montar LAN con una calidad de servicio elevada y es una buena opción cuando deben montarse redes de alta velocidad y baja latencia, como por ejemplo aquellas que involucren distribución de vídeo en tiempo real.

Existe otro hardware soportado por GNU/Linux para la interconexión de ordenadores, entre los cuales podemos mencionar: Frame Relay o X.25 (utilizada en ordenadores que acceden o interconectan WAN y para servidores con grandes necesidades de transferencias de datos), Packet Radio (interconexión vía radio utilizando protocolos como AX.25, NetRom o Rose) o dispositivos *dialing up*, que utilizan líneas series, lentas pero muy baratas, a través de módems analógicos o digitales (RDSI, DSL, ADSL, etc.). Estas últimas son las que comúnmente se utilizan en pymes o uso doméstico y requieren otro protocolo para la transmisión de paquetes, tal como SLIP o PPP. Para virtualizar la diversidad de hardware sobre una red, TCP/IP define una interfaz abstracta mediante la cual se concentrarán todos los paquetes que serán enviados por un dispositivo físico (lo cual también significa una red o un segmento de esta red). Por ello, por cada dispositivo de comunicación en la máquina tendremos una interfaz correspondiente en el *kernel* del sistema operativo.

Ejemplo

Ethernet en GNU/Linux se llaman con ethx (donde en todas, 0 indica un número de orden comenzando por 0), la interfaz a líneas series (módems) se llaman por pppx (para PPP) o slx (para SLIP), para FDDI son fddix. Estos nombres son utilizados por los comandos para configurarlas y asignarles el número de identificación que posteriormente permitirá comunicarse con otros dispositivos en la red.

En GNU/Linux puede significar tener que incluir los módulos adecuados para el dispositivo (*network interface card* NIC) adecuado (en el *kernel* o como módulos), esto significa compilar el *kernel* después de haber escogido con, por ejemplo, *make menuconfig* el NIC adecuado, indicándole como interno o como módulo (en este último caso se deberá compilar el módulo adecuado también).

Nota

Cómo ver las interfaces de red disponibles?

`ifconfig -a`

Este comando muestra todas las interfaces/parámetros por defecto de cada una.

Los dispositivos de red se pueden mirar en el directorio /dev que es donde existe un archivo (especial, ya sea de bloque o de caracteres según su transferencia), que representa a cada dispositivo hardware.[KD00, Dra99].

2. Conceptos en TCP/IP

Como se ha observado, la comunicación significa una serie de conceptos que ampliaremos a continuación [Mal96, Com01]:

- **Internet/intranet**: el término *intranet* se refiere a la aplicación de tecnologías de Internet (red de redes) dentro de una organización básicamente para distribuir y tener disponible información dentro de la compañía. Por ejemplo, los servicios ofrecidos por GNU/Linux como servicios Internet e intranet incluyen correo electrónico, WWW, *news*, etc.

- **Nodo**: se denomina nodo (*host*) a una máquina que se conecta a la red (en un sentido amplio un nodo puede ser un ordenador, una impresora, una torre (*rack*) de CD, etc.), es decir, un elemento activo y diferenciable en la red que reclama o presta algún servicio y/o comparte información.

- **Dirección de red Ethernet** (*Ethernet address* o *MAC address*): un número de 48 bits (por ejemplo 00:88:40:73:AB:FF –en octal– 0000 0000 1000 1000 0100 0000 0111 0011 1010 1011 1111 1111 –en binario–) que se encuentra en el dispositivo físico (hardware) del controlador (NIC) de red Ethernet y es grabado por el fabricante del mismo (este número debe ser único en el mundo, por lo que cada fabricante de NIC tiene un rango preasignado).

Nota

Nombre de la máquina:
`more /etc/hostname`

- **Host name**: cada nodo debe tener además un único nombre en la red. Ellos pueden ser sólo nombres o bien utilizar un esquema de nombres jerárquico basado en dominios (*hierarchical domain naming scheme*). Los nombres de los nodos deben ser únicos, lo cual resulta fácil en pequeñas redes, más dificultoso en redes extensas e imposible en Internet si no se realiza algún control. Los nombres deben ser de un máximo de 32 caracteres entre a-zA-Z0-9.-, y que no contengan espacios o # comenzando por un carácter alfabético.

Nota

Dirección IP de la máquina:
`more /etc/hosts`

- **Dirección de Internet** (*IP address*): está compuesto por cuatro números en el rango 0-255 separados por puntos (por ejemplo 192.168.0.1) y es utilizado universalmente para identificar los ordenadores sobre una red o Internet. La traslación de nombres en direcciones IP es realizada por un servidor DNS (*domain name system*) que transforma los nombres de nodo (legibles por humanos) en direcciones IP (este servicio es realizado por una aplicación denominada *named*).

Nota

Puertos preasignados en UNIX:
`more /etc/services`
Este comando muestra los puertos predefinidos por orden y según soporten TCP o UDP.

- **Puerto** (*port*): identificador numérico del buzón en un nodo que permite que un mensaje (TCP, UDP) pueda ser leído por una aplicación concreta dentro de este nodo (por ejemplo, dos máquinas que se comuniquen por telnet lo harán por el puerto 23, pero las dos mismas máquinas pueden tener una comunicación ftp por el puerto 21). Se pueden tener diferentes

Nota

Visualización de la configuración del routing:
`netstat -r`

aplicaciones comunicándose entre dos nodos a través de diferentes puertos simultáneamente.

- **Nodo *router*** (*gateway*): es un nodo que realiza encaminamientos (transferencia de datos *routing*). Un *router*, según sus características, podrá transferir información entre dos redes de protocolos similares o diferentes y puede ser además selectivo.

Nota

Dominio y quién es nuestro servidor de DNS:

```
more /etc/default domain
more /etc/resolv.conf
```

Nota

Tablas de arp:

```
arp a NombreNodo
```

- ***Domain name system*** (DNS): permite asegurar un único nombre y facilitar la administración de las bases de datos que realizan la traslación entre nombre y dirección de Internet y se estructuran en forma de árbol. Para ello, se especifican dominios separados por puntos, de los que el más alto (de derecha a izquierda) describe una categoría, institución o país (COM, comercial, EDU, educación, GOV, gubernamental, MIL, militar (gobierno), ORG, sin fin de lucro, XX dos letras por país, o casos especiales tres letras CAT lengua y cultura catalana...). El segundo nivel representa la organización, el tercero y restantes departamentos, secciones o divisiones dentro de una organización (por ejemplo, www.uoc.edu o nteum@pirulo.remix.es). Los dos primeros nombres (de derecha a izquierda, *uoc.edu* en el primer caso, *remix.es* (en el segundo) deben ser asignados (aprobados) por el SRI-NIC (órgano mundial gestor de Internet) y los restantes pueden ser configurados/asignados por la institución.

- **DHCP, bootp**: DHCP y bootp son protocolos que permiten a un nodo cliente obtener información de la red (tal como la dirección IP del nodo). Muchas organizaciones con gran cantidad de máquinas utilizan este mecanismo para facilitar la administración en grandes redes o donde existe una gran cantidad de usuarios móviles.

- **ARP, RARP:** en algunas redes (como por ejemplo IEEE 802 LAN que es el estándar para Ethernet), las direcciones IP son descubiertas automáticamente a través de dos protocolos miembros de *Internet protocol suite: address resolution protocol* (ARP) y *reverse address resolution protocol* (RARP). ARP utiliza mensajes (*broadcast messages*) para determinar la dirección Ethernet (especificación MAC de la capa 3 del modelo OSI) correspondiente a una dirección de red particular (IP). RARP utiliza mensajes de tipo *broadcast* (mensaje que llega a todos los nodos) para determinar la dirección de red asociada con una dirección hardware en particular. RARP es especialmente importante en máquinas sin disco, en las cuales la dirección de red generalmente no se conoce en el momento del inicio (*boot*).

- **Biblioteca de *sockets***: en UNIX toda la implementación de TCP/IP forma parte del *kernel* del sistema operativo (o bien dentro del mismo o como un módulo que se carga en el momento del inicio como el caso de GNU/Linux con los controladores de dispositivos).

La forma de utilizarlas por un programador es a través de la API (*application programming interface*) que implementa ese operativo. Para TCP/IP la API más común es la Berkeley Socket Library (Windows utiliza una librería equivalente que se llama Winsocks). Esta biblioteca permite crear un punto de comunicación (*socket*), asociar éste a una dirección de un nodo remoto/puerto (*bind*) y ofrecer el servicio de comunicación (a través de *connect*, *listen*, *accept, send, sendto, recv, recvfrom,* por ejemplo). La biblioteca provee además de la forma más general de comunicación (familia AF INET) comunicaciones más optimizadas para casos que los procesos que se comunican en la misma máquina (familia AF UNIX). En GNU/Linux, la biblioteca de *socket* es parte de la biblioteca estándar de C, Libc, (Libc6 en las versiones actuales), y soporta AF_INET, AF_UNIX, AF_IPX (para protocolos de redes Novell), AF_X25 (para el protocolo X.25), AF_ATMPVC-AF_ATMSVC (para el protocolo ATM) y AF_AX25, F_NETROM, AF_ROSE (para el *amateur radio protocol*).

3. ¿Cómo se asigna una dirección Internet?

Esta dirección es asignada por el NIC y tiene dos campos. El izquierdo representa la identificación de la red y el derecho la identificación del nodo. Considerando lo mencionado anteriormente (cuatro números entre 0-255, o sea 32 bits o cuatro bytes), cada byte representa o bien la red o bien el nodo. La parte de red es asignada por el NIC y la parte del nodo es asignada por la institución o el proveedor).

Existen algunas restricciones: **0** (por ejemplo, 0.0.0.0) en el campo de red está reservado para el *routing* por defecto y **127** (por ejemplo, 127.0.0.1) está reservado para la autorreferencia (*local loopback* o *local host*), **0** en la parte de nodo se refiere a esta red (por ejemplo, 192.168.0.0) y **255** está reservado para paquetes de envío a todas las máquinas (*broadcast*) (por ejemplo, 198.162.255.255). En las diferentes asignaciones se puede tener diferentes tipos de redes o direcciones:

Clase A (*red.host.host.host*): 1.0.0.1 a 126.254.254.254 (126 redes, 16 millones de nodos) definen las grandes redes. El patrón binario es: **0** + 7 bits red + 24 bits de nodos.

Clase B (*red.red.host.host*): 128.1.0.1 a 191.255.254.254 (16K redes, 65K nodos) generalmente se utiliza el primer byte de nodo para identificar subredes dentro de una institución). El patrón binario es **10** + 14 bits de red + 16 bits de nodos.

Clase C (*red.red.red.host*): 192.1.1.1 a 223.255.255.254 (2 millones de redes, 254 de nodos). El patrón binario es **110** + 21 bits red + 8 bits de nodos.

Clase D y E (*red.red.red.host*): 224.1.1.1 a 255.255.255.254 reservado para *multicast* (desde un nodo a un conjunto de nodos que forman parte de un grupo) y propósitos experimentales.

Algunos rangos de direcciones han sido reservados para que no correspondan a redes públicas, sino a redes privadas (máquinas que se conectan entre ellas sin tener conexión con el exterior) y los mensajes no serán encaminados a través de Internet, lo cual es comúnmente conocido como intranet). Éstas son para la **clase A** 10.0.0.0 hasta 10.255.255.255, **clase B** 172.16.0.0 hasta 172.31.0.0 y **clase C** 192.168.0.0 hasta 192.168.255.0.

La dirección de *broadcast* es especial, ya que cada nodo en una red escucha todos los mensajes (además de su propia dirección). Esta dirección permite que

datagramas (generalmente información de *routing* y mensajes de aviso) puedan ser enviados a una red y todos los nodos del mismo segmento de red los puedan leer. Por ejemplo, cuando ARP busca encontrar la dirección Ethernet correspondiente a una IP, éste utiliza un mensaje de *broadcast*, el cual es enviado a todas las máquinas de la red simultáneamente. Cada nodo en la red lee este mensaje y compara la IP que se busca con la propia y le retorna un mensaje al nodo que hizo la pregunta si hay coincidencia.

Dos conceptos complementarios a lo descrito anteriormente es el de **subredes** y ***routing*** entre ellas. **Subredes** significa subdividir la parte del nodo en pequeñas redes dentro de la misma red para, por ejemplo, mejorar el tráfico. Una subred toma la responsabilidad de enviar el tráfico a ciertos rangos de direcciones IP extendiendo el mismo concepto de redes A, B, C, pero sólo aplicando esta redirección en la parte nodo de la IP. El número de bits que son interpretados como identificador de la subred es dado por una máscara de red (*netmask*) que es un número de 32 bits (igual que la IP). Para obtener el identificador de la subred, se deberá hacer una operación lógica Y (*AND*) entre la máscara y la IP, lo cual dará la IP de la subred. Por ejemplo, sea una institución que tiene una red clase B con número 172.17.0.0, y su *netmask* es, por lo tanto, 255.255.0.0. Internamente, esta red está formada por pequeñas redes (una planta del edificio por ejemplo). Así, el rango de direcciones es reasignado en 20 subnets (plantas para nosotros) 172.17.1.0 hasta 172.17.20.0. El punto que conecta todas estas plantas (*backbone*) tiene su propia dirección, por ejemplo 172.17.1.0.

Estas subredes comparten el mismo IP de red, mientras que el tercero es utilizado para identificar cada una de las subredes dentro de ella (por eso se utilizará una máscara de red 255.255.255.0).

El segundo concepto, ***routing***, representa el modo en que los mensajes son enviados a través de las subredes. Por ejemplo, sean tres departamentos con subredes Ethernet:

1. Compras (subred 172.17.2.0),
2. Clientes (subred 172.17.4.0),
3. Recursos humanos, RR.HH., (subred 172.17.6.0)
4. *Backbone* con FFDI (subred 172.17.1.0).

Para encaminar los mensajes entre los ordenadores de las tres redes, se necesitarán tres *gateways* que tendrán cada uno dos interfaces de red para cambiar entre Ethernet y FFDI. Éstas serán:

1. CromprasGW IPs:172.17.2.1 y 172.17.1.1,
2. ClientesGW IPs:172.17.4.1 y 172.17.1.2
3. RRHHGW IPs:172.17.6.1 y 172.17.1.3, es decir, una IP hacia el lado de la subnet y otra hacia el *backbone*.

Cuando se envían mensajes entre máquinas de compras, no es necesario salir al *gateway*, ya que el protocolo TCP/IP encontrará la máquina directamente. El problema está cuando la máquina *Compras0* quiere enviar un mensaje a RRHH3. El mensaje debe circular por los dos *gateways* respectivos. Cuando Compras0 "ve" que RRHH3 está en otra red, envía el paquete a través del *gateway* ComprasGW, que a su vez se lo enviará a RRHHGW y que a su vez se lo enviará a RRHH3. La ventaja de las subredes es clara, ya que el tráfico entre todas las máquinas de compras, por ejemplo, no afectará a las máquinas de clientes o RR.HH. (si bien significa un planteamiento más complejo y caro a la hora de diseñar, y construir la red).

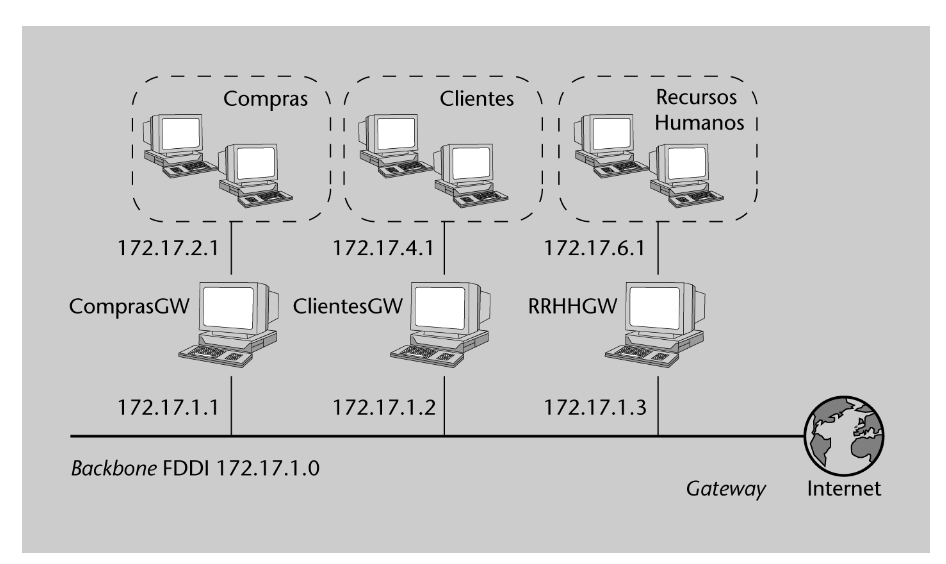


Figura 1. Configuración de segmentos y *gateways* en una Intranet

IP utiliza una tabla para hacer el *routing* de los paquetes entre las diferentes redes y en la cual existe un *routing* por defecto asociado a la red 0.0.0.0. Todas las direcciones que coinciden con ésta, ya que ninguno de los 32 bits son necesarios, son enviadas por el *gateway* por defecto (*default gateway*) hacia la red indicada. Sobre comprasGW, por ejemplo, la tabla podría ser:

Dirección	Máscara	Gateway	Interfaz
172.17.1.0	255.255.255.0	-	fddi0
172.17.4.0	255.255.255.0	172.17.1.2	fddi0
172.17.6.0	255.255.255.0	172.17.1.3	fddi0
0.0.0.0	0.0.0.0	172.17.2.1	fddi0
172.17.2.0	255.255.255.0	-	eth0

El '-' significa que la máquina está directamente conectada y no necesita *routing*. El procedimiento para identificar si se realiza el *routing* o no, se lleva a cabo a través de una operación muy simple con dos AND lógicos (*subred* AND *mask* y *origen* AND *mask*) y una comparación entre los dos resultados. Si son iguales no hay *routing*, sino que se debe enviar la máquina definida como *gateway* en cada máquina para que ésta realice el *routing* del mensaje.

Por ejemplo, un mensaje de la 172.17.2.4 hacia la 172.17.2.6 significará:

172.17.2.4 AND 255.255.255.0 = 172.17.2.0
172.17.2.6 AND 255.255.255.0 = 172.17.2.0

Como los resultados son iguales, no habrá *routing*. En cambio, si hacemos lo mismo con 172.17.2.4 hacia 172.17.6.6 podemos ver que habrá un *routing* a través del 172.17.2.1 con un cambio de interfaz (*eth0* a *ffdi0*) a la 172.17.1.1 y de ésta hacia la 172.17.1.2 con otro cambio de interfaz (*fddi0* a *eth0*) y luego hacia la 172.17.6.6. El *routing*, por defecto, se utilizará cuando ninguna regla satisfaga la coincidencia. En caso de que dos reglas coincidan, se utilizará aquella que lo haga de modo más preciso, es decir, la que menos ceros tenga. Para construir las tablas de *routing*, se puede utilizar el comando *route* durante el arranque de la máquina, pero si es necesario utilizar reglas más complejas (o *routing* automático), se puede utilizar el *routing information protocol* (RIP) o entre sistemas autónomos el *external gateway protocol* (EGP) o también el *border gateway protocol* (BGP). Estos protocolos se implementan en el comando *gated*.

Para instalar una máquina sobre una red existente, es necesario, por lo tanto, disponer de la siguiente información obtenida del proveedor de red o de su administrador: dirección IP del nodo, dirección de la red IP, dirección de *broadcast*, dirección de máscara de red, dirección de *router*, dirección del DNS.

Si se construye una red que nunca tendrá conexión a Internet, se pueden escoger las direcciones que se prefieran, pero es recomendable mantener un orden adecuado en función del tamaño de red que se desee tener y para evitar problemas de administración dentro de dicha red. A continuación, se verá cómo se define la red y el nodo para una red privada (hay que ser cuidadoso, ya que si se tiene la máquina conectada a la red, se podría perjudicar a otro usuario que tuviera asignada esta dirección): dirección de nodo 192.168.110.23, máscara de red 255.255.255.0, parte de red 192.168.110., parte de nodo .23, dirección de red 192.168.110.0, dirección de *broadcast* 192.168.110.255.

4. ¿Cómo se debe configurar la red?

4.1. Configuración de la interfaz (NIC, *network interface controller*)

Una vez cargado el *kernel* de GNU/Linux, éste ejecuta el comando init que a su vez lee el archivo de configuración /etc/inittab y comienza el proceso de inicialización. Generalmente, el inittab tiene secuencias tales como: si::sysinit: /etc/init.d/boot, que representa el nombre del archivo de comandos (*script*) que controla las secuencias de inicialización. Generalmente este *script* llama a otros *scripts*, entre los cuales se encuentra la inicialización de la red.

Ejemplo

En Debian se ejecuta etc/init.d/network para la configuración de la interfaz de red y en función del nivel de arranque; por ejemplo, en el 2 se ejecutarán todos los ficheros S* del directorio /etc/rc2.d (que son enlaces al directorio /etc/initd), y en el nivel de apagado, todos los K* del mismo directorio. De este modo, el *script* está sólo una vez (/etc/init.d) y de acuerdo a los servicios deseados en ese estado se crea un enlace en el directorio correspondiente a la configuración del nodo-estado.

Los dispositivos de red se crean automáticamente cuando se inicializa el hardware correspondiente. Por ejemplo, el controlador de Ethernet crea las interfaces eth[0..n] secuencialmente cuando se localiza el hardware correspondiente.

Nota

Consultar
`man ifconfig`
para las diferentes opciones del comando.

A partir de este momento, se puede configurar la interfaz de red, lo cual implica dos pasos: asignar la dirección de red al dispositivo e inicializar los parámetros de la red al sistema. El comando utilizado para ello es el ifconfig (*interface configure*). Un ejemplo será:

ifconfig eth0 192.168.110.23 netmask 255.255.255.0 up

Lo cual indica configurar el dispositivo eth0 con dirección IP 192.168.110.23 y máscara de red 255.255.255.0. El *up* indica que la interfaz pasará al estado activo (para desactivarla debería ejecutarse ifconfig eth0 down). El comando asume que si algunos valores no se indican, son tomados por defecto. En este caso, el *kernel* configurará esta máquina como Tipo-C y configurará la red con 192.168.110.23 y la dirección de *broadcast* con 192.168.110.255. Por ejemplo:

ifconfig eth0 192.168.110.23 netmask 255.255.255.0 up

Existen comandos como el ifup e ifdown, que permite configurar/desconfigurar la red en forma más simple utilizando el archivo /etc/network/interfaces para obtener todos los parámetros necesarios (consultar *man interfaces* para su sintaxis).

En Debian, con el fin de facilitar la configuración de la red, existe otra forma de configurar la red (considerada de alto nivel) que utiliza los comandos mencionados anteriormente ifup, ifdown y el archivo /etc/network/interfaces. Si

se decide utilizar estos comando **no** se debería configurar la red a bajo nivel, ya que estos comandos son suficientes para configurar/desconfigurar la red.

Para modificar los parámetros de red de la interfaz eth0 se puede hacer (consultar *man interfaces* en la sección 5 del manual de Unix incluido con el sistema operativo para más información del formato):

ifdown eth0	para todos los servicios de red sobre eth0
vi /etc/network/interfaces	edite y modifique los que necesite
ifup eth0	pone en marcha los servicios de red sobre eth0

Supongamos que desea configurar sobre Debian una interfaz eth0 que tiene una dirección IP fija `192.168.0.123`. y con `192.168.0.1` como puerta de enlace (*gateway*). Se debe editat /etc/network/interfaces de modo que incluya una sección como:

```
iface eth0 inet static
        address 192.168.0.123
        netmask 255.255.255.0
        gateway 192.168.0.1
```

Si tiene instalado el paquete `resolvconf` puede añadir líneas para especificar la información relativa al DNS. Por ejemplo:

```
iface eth0 inet static
        address 192.168.0.123
        netmask 255.255.255.0

        gateway 192.168.0.1
        dns-search remix.org
        dns-nameservers 195.238.2.21 195.238.2.22
```

Después de activarse la interfaz, los argumentos de las opciones dns-search y dns-nameservers quedan disponibles para resolvconf para su inclusión en resolv.conf. El argumento remix.org de la opción dns-search corresponde al argumento de la opción search en resolv.conf (se verá más adelante) y los argumentos 195.238.2.21 y 195.238.2.22 de la opción dns-nameservers corresponde a los argumentos de las opciones nameserver en resolv.conf(consultar man resolv.conf). También se puede configurar la red a bajo nivel a través del comando ip (que es equivalente a ifconfig y route). Si bien este comando es mucho más versátil y potente (permite establecer túneles, routing alternativos, etc) es más complejo y se recomienda utilizar los procedimientos anteriores para configuraciones básicas de la red.

4.1.1. Configuración de red en (estilo) Fedora

Red Hat y Fedora utilizan diferente estructura de ficheros para la configuración de la red: /etc/sysconfig/network. Por ejemplo para la configuración estática de la red:

NETWORKING=yes
HOSTNAME=*my-hostname* *Nombre del host definido por el cmd hostname*
FORWARD_IPV4=true *True para NAT firewall gateways y routers.*
False para cualquier otro caso
GATEWAY="*XXX.XXX.XXX.YYY*" *Puerta de salida a Internet*

Para configuración por DHCP se debe quitar la línea de GATEWAY, ya que será asignada por el servidor. Y en caso de incorporar NIS debe agregarse una línea con el servidor de dominio: NISDOMAIN=NISProject1

Para configurar la interfaz eth0 en el archivo

/etc/sysconfig/network-scripts/ifcfg-eth0:

DEVICE=eth0
BOOTPROTO=static
BROADCAST=XXX.XXX.XXX.255
IPADDR=*XXX.XXX.XXX.XXX*
NETMASK=255.255.255.0
NETWORK=*XXX.XXX.XXX.0*
ONBOOT=yes Activará la red en el boot.

También a partir de FC3 se pueden agregar:
TYPE=Ethernet
HWADDR=*XX:XX:XX:XX:XX:XX*
GATEWAY=*XXX.XXX.XXX.XXX*
IPV6INIT=no
USERCTL=no
PEERDNS=yes

O si no para configuración por DHCP :

DEVICE=eth0
ONBOOT=yes
BOOTPROTO=dhcp

Para deshabilitar DCHP, cambiar BOOTPROTO=dhcp a BOOTPROTO=none. Cualquier cambio en estos ficheros deberá reiniciar los servicios con service network restart (o sino /etc/init.d/network restart).

Para cambiar el nombre del *host*, se deben seguir estos tres pasos:

1) El comando `hostname nombre-nuevo.`

2) Cambiar la configuración de la red en /etc/sysconfig/network editando HOSTNAME=nombre-nuevo.

3) Restaurando los servicios (o haciendo un *reboot*):

- `service network restart` (o: /etc/init.d/network restart)

- Reiniciando el desktop pasando a modo consola `init 3` y cambiando a modo GUI `init 5`.

Verificar si el nombre también no está dado de alta en el /etc/hosts. El hostname puede ser cambiado en tiempo de ejecución con `sysctl` -w kernel.hostname="nombre-nuevo".

4.1.2. Configuración de un red Wi-Fi (inalámbrica)

Para la configuración de interfaces Wi-Fi se utiliza básicamente el paquete wireless-tools (además de ifconfig o ip). Este paquete utiliza el comando iwconfig para configurar una interfaz inalámbrica, pero también se puede hacer a través del /etc/network/interfaces.

Ejemplo: Configurar una WiFi en Debian Sarge(Etch) (similar en FC6)

Supongamos que queremos configurar una tarjeta de red inalámbrica Intel Pro/Wireless 2200BG (muy común en una gran cantidad de portátiles -p. ej. Dell, HP...-). Normalmente el software que controla las tarjetas se divide en dos partes: el módulo software que se cargará en el *kernel* a través del comando modprobe y el firmware que es el código que se cargará en la tarjeta y que nos da el fabricante (consultar la página de Intel para este modelo). Como estamos hablando de módulos, es interesante utilizar el paquete de debian module-assistant que nos permite crear e instalar fácilmente un módulo (otra opción sería instalar las fuentes y crear el módulo correspondiente). El software (lo encontramos en la página del fabricante y lo denomina ipw2200) lo compilaremos e instalaremos con el comando m-a del paquete module-assistant.

```
aptget install module-assistant (instalo el paquete)
m-a -t update
m-a -t -f get ipw2200
m-a -t -build ipw2200
m-a -t install ipw2200
```

Desde la dirección indicada por el fabricante (en su documentación) se descarga la versión del firmaware compatible con la versión del *driver*, en nuestro caso para el *driver* versión 1.8 lel firmware es la 2.0.4 obtenida desde la página:

http://ipw2200.sourceforge.net/firmware.php

Y a continuación se descomprime e instala el firmware:

```
tar xzvf ipw2200fw2.4.tgz C /tmp/fwr/
cp /tmp/fwr/*.fw /usr/lib/hotplug/firmware/
```

Con esto se copiarán tres paquetes (ipw2200-bss.fw, ipw2200-ibss.fw y ipw2200-sniffer.fw). Luego se carga el módulo con: `modprobe ipw2200`, se reinicia el sistema (reboot) y luego desde consola podemos hacer `dmesg | grep ipw`, este comando nos mostrará algunas líneas similares a las que se muestran a continuación y que indicarán que el módulo está cargado (se puede verificar con lsmod):

```
ipw2200: Intel(R) PRO/Wireless 2200/2915 Network Driver, git1.0.8
ipw2200: Detected Intel PRO/Wireless 2200BG Network Connection
...
```

Luego se descarga el paquete wirelesstools que contiene iwconfig y entre otras con aptget install wirelesstools y ejecutamos iwconfig y saldrá algo parecido a:

```
eth1 IEEE 802.11b ESSID:"Nombre-de-la-Wifi"
Mode:Managed Frequency:2.437 GHz
Access Point:00:0E:38:84:C8:72
Bit Rate=11 Mb/s TxPower=20 dBm
Security mode:open
   ...
```

Se debe a continuación configurar el archivo de redes por ejemplo `gedit /etc/network/interfaces` y añadir la interfaz wifi eth1 por ejemplo:

```
iface eth1 inet dhcp
      pre-up iwconfig eth1 essid "Nombre de la Wifi"
      pre-up iwconfig eth1 key open XXXXXXXXXX
```

La líneas pre-up ejecuta el comando iwconfig antes de activar la interfaz. Esta configuración se hace si se quiere utilizar un servicio en modo DHCP (asignación automática de IP, se verá más adelante). Se debe utilizar en vez de dhcp la palabra static y además poner las siguientes líneas por ejemplo (como en una tarjeta de cable):

```
address 192.168.1.132
netmask 255.255.255.0
network 192.168.0.0
broadcast 192.168.0.255
gateway 192.168.1.1
```

Un método alternativo para configurar la interfaz es:

```
iface eth1 inet dhcp
      wireless-essid "Nombre de la Wifi"
      wireless-key 123456789e
```

A continuación se puede poner en marcha la red con ifup eth1 y nos dará información sobre la conexión y nos indicará su estado y calidad de recepción. Para buscar (scan) las redes WiFi disponibles (puntos de acceso) podemos utilizar `iwlist scan`, lo que nos mostrará información de las redes disponibles, y si nos queremos conectar a una diferente, se puede utilizar el comando iwconfig para cambiar de red o punto de acceso (Access Point).

4.2. Configuración del Name Resolver

El siguiente paso es configurar el *name resolver* que convierte nombres tales como pirulo.remix.com en 192.168.110.23. El archivo /etc/resolv.conf es el utilizado para tal fin. Su formato es muy simple (una línea de texto por sentencia). Existen tres palabras clave para tal fin: *domain* (dominio local), *search* (lista de dominios alternativos) y *name server* (la dirección IP del *domain name server*).

Ejemplo de /etc/resolv.conf

```
domain remix.com
search remix.com piru.com
name server 192.168.110.1
name server 192.168.110.65
```

Esta lista de servidores de nombre a menudo dependen del entorno de red, que puede cambiar dependiendo de dónde esté o se conecte la máquina. Los programas de conexión a líneas telefónicas (`pppd`) o obtención de direcciones IP auto-

máticamente (dhclient) son capaces de modificar resolv.conf para insertar o eliminar servidores; pero estas características no siempre funcionan adecuadamente y a veces pueden entrar en conflicto y generar configuraciones erróneas. El paquete **resolvconf** (aún en unstable) soluciona de forma adecuada el problema y permite una configuración simple de los servidores de nombre en forma dinámica. resolvconf está diseñado para funcionar sin que sea necesaria ninguna configuración manual, no obstante, el paquete es bastante nuevo y puede requerir alguna intervención para lograr que funcione adecuadamente. Para más información:

http://packages.debian.org/unstable/net/resolvconf

Un archivo importante es el /etc/*host*.conf, que permite configurar el comportamiento del *name* resolver. Su importancia reside en indicar dónde se resuelve primero la dirección o el nombre de un nodo. Esta consulta puede hacerse al servidor DNS o a tablas locales dentro de la máquina actual (/etc/hosts).

Ejemplo de /etc/host.conf

```
order hosts,bind
multi on
```

Esta configuración indica que primero se verifique el /etc/hosts antes de solicitar una petición al DNS y también indica (2.ª línea) que retorne todas las direcciones válidas que se encuentren en /etc/hosts. Por lo cual, el archivo /etc/hosts es donde se colocan las direcciones locales o también sirve para acceder a nodos sin tener que consultar al DNS.

La consulta es mucho más rápida, pero tiene la desventaja de que si el nodo cambia, la dirección será incorrecta. En un sistema correctamente configurado, sólo deberán aparecer el nodo local y una entrada para la interfaz *loopback*.

Ejemplo de /etc/hosts

```
127.0.0.1       localhost              loopback
192.168.1.2     pirulo.remix.com       pirulo
```

Para el nombre de una máquina pueden utilizarse alias, que significa que esa máquina puede llamarse de diferentes maneras para la misma dirección IP. En referencia a la interfaz *loopback,* éste es un tipo especial de interfaz que permite realizar a nodo conexiones consigo misma (por ejemplo, para verificar que el subsistema de red funciona sin acceder a la red). Por defecto, la dirección IP 127.0.0.1 ha sido asignada específicamente al *loopback* (un comando telnet 127.0.0.1 conectará con la misma máquina). Su configuración es muy fácil (la efectúan generalmente los *script* de inicialización de red).

Ejemplo del *loopback*

```
ifconfig lo 127.0.0.1
route add host 127.0.0.1 lo
```

En la versión 2 de la biblioteca GNU existe un reemplazo importante con respecto a la funcionalidad del archivo host.conf. Esta mejora incluye la centralización de información de diferentes servicios para la resolución de nombres, lo cual presenta grandes ventajas para el administrador de red. Toda la información de consulta de nombres y servicios ha sido centralizada en el archivo /etc/nsswitch.conf, el cual permite al administrador configurar el orden y las bases de datos de modo muy simple. En este archivo cada servicio aparece uno por línea con un conjunto de opciones, donde, por ejemplo, la resolución de nombres de nodo es una de ellas. En éste se indica que el orden de consulta de las bases de datos para obtener el IP del nodo o su nombre será primero el servicio de DNS (que utilizará el archivo /etc/resolv.conf para determinar la IP del nodo DNS) y en caso de que no pueda obtenerlo, utilizará el de las bases de datos local (/etc/*hosts*). Otras opciones para ello podrían ser nis, nisplus, que son otros servicios de información que serán descritos en unidades posteriores. También se puede controlar por medio de acciones (entre []) el comportamiento de cada consulta, por ejemplo:

```
hosts: xfn nisplus dns [NOTFOUND = return] files
```

Esto indica que cuando se realice la consulta al DNS, si no existe un registro para esta consulta, retorne al programa que la hizo con un cero. Puede utilizarse el '!' para negar la acción, por ejemplo:

```
hosts dns [!UNAVAIL = return] files
```

Nota

Ejemplo de nsswitch.conf: ...
hosts: dns files
...
networks: files

4.3. Configuración del *routing*

Otro aspecto que hay que configurar es el *routing*. Si bien existe el tópico sobre su dificultad, generalmente se necesitan unos requisitos de *routing* muy simples. En un nodo con múltiples conexiones, el *routing* consiste en decidir dónde hay que enviar y qué se recibe. Un nodo simple (una sola conexión de red) también necesita *routing*, ya que todos los nodos disponen de un *loopback* y una conexión de red (por ejemplo, Ethernet, PPP, SLIP...). Como se explicó anteriormente, existe una tabla llamada *routing table* que contiene filas con diversos campos, pero con tres campos sumamente importantes: dirección de destino, interfaz por donde saldrá el mensaje, y dirección IP, que efectuará el siguiente paso en la red (*gateway*).

Nota

Consulta de tablas de routing:
`route -n`
o también
`netstat -r`

El comando route permite modificar esta tabla para realizar las tareas de *routing* adecuadas. Cuando llega un mensaje, se mira su dirección destino, se compara con las entradas en la tabla y se envía por la interfaz cuya dirección coincide mejor con el destino del paquete. Si un *gateway* es especificado, se envía a la interfaz adecuada.

Consideremos, por ejemplo, que nuestro nodo está en una red clase C con dirección 192.168.110.0 y tiene una dirección 192.168.110.23; y el *router* con conexión a Internet es 192.168.110.3. La configuración será:

- Primero la interfaz:

  ifconfig eth0 192.168.110.23 netmask 255.255.255.0 up

- Más adelante, indicar que todos los datagramas para nodo con direcciones 192.168.0.* deben ser enviados al dispositivo de red:

  route add -net 192.1 ethernetmask 255.255.255.0 eth0

El -*net* indica que es una ruta de red pero también puede utilizarse -*host* 192.168.110.3. Esta configuración permitirá conectarse a todos los nodos dentro del segmento de red (192.1), pero ¿qué pasará si se desea conectar con otro nodo fuera de este segmento? Sería muy difícil tener todas las entradas adecuadas para todas las máquinas a las cuales se quiere conectar. Para simplificar esta tarea, existe el *default route*, que es utilizado cuando la dirección destino no coincide en la tabla con ninguna de las entradas. Una posibilidad de configuración sería:

route add default gw 192.168.110.3 eth0

(el gw es la IP o nombre de un *gateway* o nodo *router*).

Una forma alternativa de hacerlo es:

`ifconfig eth0 inet down` *deshabilito la interfaz*
`ifconfig lo Link encap:Local Loopback`
... (no mostrará ninguna entradas para eth0)
`route`
... (no mostrará ninguna entrada en la tabla de rutas)

Luego se habilita la interfaz con una nueva IP y una la nueva ruta:

```
ifconfig eth0 inet up 192.168.0.111 \
        netmask 255.255.0.0 broadcast 192.168.255.255
route add -net 10.0.0.0 netmask 255.0.0.0 \
        gw 192.168.0.1 dev eth0
```

La barra (\) indica que el comando continúa en la siguiente línea. El resultado:

```
ifconfig
    eth0 Link encap:Ethernet HWaddr 08:00:46:7A:02:B0
          inet addr:192.168.0.111 Bcast: 192.168.255.255 Mask:255.255.0.0
          UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
          ...
```

```
lo Link encap:Local Loopback
        inet addr:127.0.0.1 Mask:255.0.0.0
        ...
```

route

```
Kernel IP routing table
  Destination   Gateway        Genmask Flags Metric Ref Use Iface
  192.168.0.0   *              255.255.0.0 U       0 0 0 eth0
  10.0.0.0      192.168.0.1    255.0.0.0 UG 0 0 0 eth0
```

Para más información ver los comandos ifconfig(8) y route(8).

4.4. Configuración del inetd

El siguiente paso en la configuración de red es la configuración de los servidores y servicios que permitirán a otro usuario acceder a la máquina local o a sus servicios. Los programas servidores utilizarán los puertos para escuchar las peticiones de los clientes, los cuales se dirigirán a este servicio como IP:port. Los servidores pueden funcionar de dos maneras diferentes: standalone (en el cual el servicio escucha en el puerto asignado y siempre se encuentra activo) o a través del inetd.

> El inetd es un servidor que controla y gestiona las conexiones de red de los servicios especificados en el archivo /etc/inetd.conf, el cual, ante una petición de servicio, pone en marcha el servidor adecuado y le transfiere la comunicación.

Dos archivos importantes necesitan ser configurados: /etc/services y `/etc/inetd.conf`. En el primero se asocian los servicios, los puertos y el protocolo, y en el segundo los programas servidores que responderán ante una petición a un puerto determinado. El formato de /etc/services es `name port/protocol aliases`, donde el primer campo es nombre del servicio, el segundo, el puerto donde atiende este servicio y el protocolo que utiliza, y el siguiente, un alias del nombre. Por defecto existen una serie de servicios que ya están preconfigurados. A continuación se muestra un ejemplo de /etc/services (# indica que lo que existe a continuación es un comentario):

```
tcpmux    1/tcp # TCP port service multiplexer
echo      7/tcp
echo      7/udp
discard   9/tcp        sink null
discard   9/udp sink   null
systat    11/tcp       users
...
ftp       21/tcp
ssh       22/tcp # SSH Remote Login Protocol
ssh       22/udp # SSH Remote Login Protocol
telnet    23/tcp
```

```
# 24 - private
smtp        25/tcp        mail
...
```

El archivo /etc/inetd.conf es la configuración para el servicio maestro de red (*inetd server daemon*). Cada línea contiene siete campos separados por espacios: *service socket_type proto flags user server_path server_args*, donde *service* es el servicio descrito en la primera columna de /etc/services, *socket_type* es el tipo de *socket* (valores posibles *stream*, *dgram*, *raw*, *rdm*, o *seqpacket*), *proto* es el protocolo válido para esta entrada (debe coincidir con el de /etc/services), *flags* indica la acción que tomar cuando existe una nueva conexión sobre un servicio que se encuentra atendiendo a otra conexión (*wait* le dice a *inetd* no poner en marcha un nuevo servidor o *nowait* significa que inetd debe poner en marcha un nuevo servidor). *user* será el usuario con el cual se identificará quien ha puesto en marcha el servicio, *server_path* es el directorio donde se encuentra el servidor, y *server_args* son argumentos posibles que serán pasados al servidor. Un ejemplo de algunas líneas de /etc/inetd.conf es (recordar que # significa comentario, por lo cual, si un servicio tiene # antes de nombre, significa que no se encuentra disponible):

```
...
telnet stream tcp nowait root /usr/sbin/tcpd /usr/sbin/in.telnetd
ftp stream tcp nowait root /usr/sbin/tcpd /usr/sbin/in.ftpd
# fsp dgram udp wait root /usr/sbin/tcpd /usr/sbin/in.fspd
shell stream tcp nowait root /usr/sbin/tcpd /usr/sbin/in.rshd
login stream tcp nowait root /usr/sbin/tcpd /usr/sbin/in.rlogind
# exec stream tcp nowait root /usr/sbin/tcpd /usr/sbin/in.rexecd...
...
```

A partir de Debian Woody 3.0 r1, la funcionalidad de inetd ha sido reemplazada por xinetd (recomendable), el cual necesita el archivo de configuración /etc/xinetd.conf (ver al final de la unidad). Si se desea poner en marcha el servicio de *inetd*, se debe ejecutar (y crear los *links* adecuados en los directorios /etc/rcX.d) /etc/init.d/inetd.real start (ver al final del capítulo ejemplo de configuraciones).

Además de la configuración de inetd o xinetd, la configuración típica de los servicios de red en un entorno de escritorio o servidor básico podría incluir además (muchos de estos servicios se verán en el capítulo de servidores):

- **ssh**: conexión interactiva segura como reemplazo de telnet y incluye dos archivos de configuración /etc/ssh/ssh_config (para el cliente) y /etc/ssh/sshd_config (para el servidor)

- **exim**: agente de transporte de correo (MTA), incluye los archivos de configuración: /etc/exim/exim.conf, /etc/mailname, /etc/aliases, /etc/email-addresses.

- **fetchmail**: daemon para descargar el correo de una cuenta POP3, `/etc/fetchmailrc`
- **procmail**: programa para filtrar y distribuir el correo local, ~/.procmailrc
- **tcpd**: Servicios de filtros de máquinas y dominios habilitados y deshabilitados para conectarse al servidor (wrappers): /etc/hosts.allow, /etc/hosts.deny
- **DHCP**. Servicio para la gestión (servidor) u obtención de IP (cliente), /etc/dhcp3/dhclient.conf (cliente), /etc/default/dhcp3-server (servidor), /etc/dhcp3/dhcpd.conf (servidor)
- **CVS**: sistema de control de versiones concurrentes, /etc/cvs-cron.conf, /etc/cvs-pserver.conf
- **NFS**: sistema de archivos de red, /etc/exports
- **Samba**: sistema de archivos de red y compartición de impresoras en redes Windows, /etc/samba/smb.conf
- **lpr**: daemon para el sistema de impresión, /etc/printcap (para el sistema lpr -no para CUPS-)
- **Apache** y **Apache2:** Servidor de Web, /etc/apache/* y /etc/apache2/*
- **squid**: Servidor proxy-caché, /etc/squid/*

4.5. Configuración adicional: *protocols* y *networks*

Existen otros archivos de configuración que en la mayoría de los casos no se utilizan pero que pueden ser interesantes. El /etc/protocols es un archivo que relaciona identificadores de protocolos con nombres de protocolos, así, los programadores pueden especificar los protocolos por sus nombres en los programas.

Ejemplo de /etc/protocols

```
ip        0   IP       # internet protocol, pseudo protocol number
#hopopt   0   HOPOPT   # IPv6 Hop-by-Hop Option [RFC1883]
icmp      1   ICMP     # internet control message protocol
```

El archivo /etc/networks tiene una función similar a /etc/hosts, pero con respecto a las redes, indica nombres de red en relación con su dirección IP (el comando route mostrará el nombre de la red y no su dirección en este caso).

Ejemplo de /etc/networks

```
loopnet 127.0.0.0
localnet 192.168.0.0
amprnet 44.0.0.0 ...
```

4.6. Aspectos de seguridad

Es importante tener en cuenta los aspectos de seguridad en las conexiones a red, ya que una fuente de ataques importantes se produce a través de la red. Ya se hablará más sobre este tema en la unidad correspondiente a seguridad; sin embargo, hay unas cuantas recomendaciones básicas que deben tenerse en cuenta para minimizar los riesgos inmediatamente antes y después de configurar la red de nuestro ordenador:

• No activar servicios en /etc/inetd.conf que no se utilizarán, insertar un # antes del nombre para evitar fuentes de riesgo.

• Modificar el archivo /etc/ftpusers para denegar que ciertos usuarios puedan tener conexión vía ftp con su máquina.

• Modificar el archivo /etc/securetty para indicar desde qué terminales (un nombre por línea), por ejemplo: tty1 tty2 tty3 tty4, se permite la conexión del superusuario (*root*). Desde las terminales restantes, *root* no podrá conectarse.

• Utilizar el programa tcpd. Este servidor es un *wrapper* que permite aceptar-negar un servicio desde un determinado nodo y se coloca en el /etc/inetd.conf como intermediario de un servicio. El tcpd verifica unas reglas de acceso en dos archivos: /etc/hosts.allow /etc/host.deny.

Si se acepta la conexión, pone en marcha el servicio adecuado pasado como argumento (por ejemplo, la línea del servicio de ftp antes mostrada en inetd.conf:

ftp stream tcp nowait *root* /usr/sbin/tcpd/usr/sbin/in.ftpd.

tcpd primero busca /etc/hosts.allow y luego /etc/hosts.deny. El archivo hosts.deny contiene las reglas de cuáles son los nodos que no tienen acceso a un servicio dentro de esta máquina. Una configuración restrictiva es ALL: ALL, ya que sólo se permitirá el acceso a los servicios desde los nodos declarados en /etc/hosts.allow.

El archivo /etc/hosts.equiv permite el acceso a esta máquina sin tener que introducir una clave de acceso (*password*). Se recomienda no utilizar este mecanismo y aconsejar a los usuarios no utilizar el equivalente desde la cuenta de usuario a través del archivo .rhosts.

En Debian es importante configurar /etc/security/access.conf, el archivo que indica las reglas de quién y desde dónde se puede conectar (*login*) a esta máquina. Este archivo tiene una línea por orden con tres campos separados por ':' del tipo permiso: usuarios: origen. El primero será un +o- (acceso o denegado), el segundo un nombre de usuario/s, grupo o user@host, y el tercero un

nombre de un dispositivo, nodo, dominio, direcciones de nodo o de redes, o ALL.

Ejemplo de access.conf

Este comando no permite *root logins* sobre *tty1*:

ALL EXCEPT root:tty1...

Permite acceder a *u1*, *u2*, *g1* y todos los de dominio remix.com:

+:u1 u2 g1 .remix.com:ALL

4.7. Opciones del IP

Existen una serie de opciones sobre el tráfico IP que es conveniente mencionar. Su configuración se realiza a través de inicializar el archivo correspondiente en directorio /proc/sys/net/ipv4/. El nombre del archivo es el mismo que el del comando y para activarlos se debe poner un 1 dentro del archivo, 0 para desactivarlo.

Ejemplo

Por ejemplo, si se quiere activar ip_forward, se debería ejecutar:

echo 1 > /proc/sys/net/ipv4/ip_forward

Los más utilizados son: ip_forward utilizado para el *routing* entre interfaces o con IP Masquerading; ip_default_ttl, que es el tiempo de vida para un paquete IP (64 milisegundos por defecto) ip_bootp_agent variable lógica (BOOLEAN) que acepta paquetes (o no) con dirección origen del tipo 0.b.c.d y destino de este nodo, broadcast o multicast.

4.7.1. Comandos para la solución de problemas con la red

Si hay problemas en la configuración de la red, se puede comenzar verificando la salida de los siguientes comandos para obtener una primera idea:

```
ifconfig
cat /proc/pci
cat /proc/interrupts
dmesg | more
```

Para verificar la conexión a la red se pueden utilizar los siguientes comandos (debe tener instalado netkit-ping, traceroute, dnsutils, iptables y net-tools):

```
ping uoc.edu       # verificar la conexión a Internet
traceroute uoc.edu # rastrear paquetes IP
ifconfig           # verificar la configuración del host
route -n           # verificar la configuración de la ruta
```

```
dig [@dns.uoc.edu] www.uoc.edu # verificar registros de www.uoc.edu
                              # sobre el servidor dns.uoc.edu
iptables -L -n |less          # verificar filtrado de paquetes (kernel >=2.4)
netstat -a                    # muestra todos los puertos abiertos
netstat -l --inet             # muestra los puertos en escucha
netstat -ln --tcp             # mostrar puertos tcp en escucha (numérico)
```

5. Configuración del DHCP

DHCP son las siglas de *dynamic host configuration protocol*. Su configuración es muy simple y sirve para que, en lugar de configurar cada nodo de una red individualmente, se pueda hacer de forma centralizada y su administración sea más fácil. La configuración de un cliente es muy fácil, ya que sólo se debe instalar uno de los siguientes paquetes: `dhcp3-client` (versión 3, Internet Software Consortium), `dhcpcd` (Yoichi Hariguchi y Sergei Viznyuk), `pump` (Red Hat) y agregando la palabra *dhcp* en la entrada correspondiente a la interfaz que se desea que funcione bajo el cliente dhcp (p. ej. /etc/network/interfaces debe tener iface eth0 inet dhcp...).

La configuración del servidor requiere un poco más de atención, pero no presenta complicaciones. Primero, para que el servidor pueda servir a todos los clientes DHCP (incluido Windows), deben realizarse algunas cuestiones previas relacionadas con las direcciones de *broadcast*. Para ello, primero el servidor debe poder enviar mensajes a la dirección 255.255.255.255, lo cual no es válido en GNU/Linux. Para probarlo, ejecútese:

route add -host 255.255.255.255 dev eth0

Si aparece el siguiente mensaje *255.255.255.255: Unknown host*, debe añadirse la siguiente entrada en /etc/hosts: *255.255.255.255 dhcp* e intentar nuevamente:

route add -host dhcp dev eth0

La configuración de dhcpd se puede realizar con la interfaz gráfica de `linux-conf`, (no recomendable) o bien editar /etc/dhcpd.conf. Un ejemplo de este archivo es:

```
# Ejemplo de /etc/dhcpd.conf:
default-lease-time 1200;
max-lease-time 9200;
option domain-name "remix.com";
deny unknown-clients;
deny bootp;
option broadcast-address 192.168.11.255;
option routers 192.168.11.254;
option domain-name-servers 192.168.11.1, 192.168.168.11.2;
subnet 192.168.11.0 netmask 255.255.255.0
{ not authoritative;
        range 192.168.11.1 192.168.11.254
        host marte {
                hardware ethernet 00:00:95:C7:06:4C;
                fixed address 192.168.11.146;
                option host-name "marte";
                }
        host saturno {
                hardware ethernet 00:00:95:C7:06:44;
                fixed address 192.168.11.147;
                option host-name "saturno";
        }
}
```

Esto permitirá al servidor asignar el rango de direcciones 192.168.11.1 al 192.168.11.254 tal y como se describe cada nodo. Si no existe el segmento *host* { ... } correspondiente, se asignan aleatoriamente. Las IP son asignadas por un tiempo mínimo de 1.200 segundos y máximo de 9.200 (en caso de no existir estos parámetros, se asignan indefinidamente).

Antes de ejecutar el servidor, debe verificarse si existe el fichero `/var/state/dhcp/dhcpd.leases` (en caso contrario, habrá que crearlo con touch /var/state/dhcp/dhcpd.leases). Para ejecutar el servidor: /usr/sbin/dhcpd (o bien ponerlo en los *scripts* de inicialización). Con `/usr/sbin/dhcpd -d -f` se podrá ver la actividad del servidor sobre la consola del sistema. [Mou01, Rid00, KD00, Dra99]

Es importante no olvidar la sentencia `not autoritative` ya que sino este servidor puede dejar sin funcionamiento a otros servidores de dhcp que sirvan IP de otros segmentos.

6. IP *aliasing*

Existen algunas aplicaciones donde es útil configurar múltiples direcciones IP a un único dispositivo de red. Los ISP (*Internet service providers*) utilizan frecuentemente esta característica para proveer de características personalizadas (por ejemplo, de World Wide Web y FTP) a sus usuarios. Para ello, el *kernel* debe estar compilado con las opciones de Network Aliasing e IP (*aliasing support*). Después de instalado el nuevo *kernel*, la configuración es muy fácil. Los alias son anexados a dispositivos de red virtuales asociados con el nuevo dispositivo con un formato tal como: dispositivo: número virtual.

Por ejemplo: eth0:0, ppp0:8

Consideremos que tenemos una red Ethernet que soporta dos diferentes subredes IP simultáneamente y que nuestra máquina desea tener acceso directo a ellas. Un ejemplo de configuración sería:

```
ifconfig eth0 192.168.110.23 netmask 255.255.255.0 up
route add -net 192.168.110.0 netmask 255.255.255.0 eth0
ifconfig eth0:0 192.168.10.23 netmask 255.255.255.0 up
route add -net 192.168.10.0 netmask 255.255.255.0 eth0:0
```

Lo cual significa que tendremos dos IP 192.168.110.23 y 192.168.10.23 para la misma NIC. Para borrar un alias, agregar un '-' al final del nombre (por ejemplo, ifconfig eth0:0- 0). [Mou01, Ran05]

Un caso típico es que se desee configurar una única tarjeta Ethernet para que sea la interfaz de distintas subredes IP. Por ejemplo, supongamos que se tiene una máquina que se encuentra en una red LAN 192.168.0.x/24. Y se desea conectar la máquina a Internet usando una dirección IP pública proporcionada con DHCP usando su tarjeta Ethernet existente. Por ejemplo, se puede hacer como en el ejemplo anterior o también editar el archivo /etc/network/interfaces de modo que incluya una sección similar al siguiente:

```
iface eth0 inet static
address 192.168.0.1
netmask 255.255.255.0
network 192.168.0.0
broadcast 192.168.0.255

iface eth0:0 inet dhcp
```

La interfaz eth0:0 es una interfaz virtual y al activarse también lo hará su padre eth0.

7. IP Masquerade

El IP Masquerade es un recurso para que un conjunto de máquinas puedan utilizar una única dirección IP. Esto permite que los nodos ocultos (es decir, los que utilizan una IP privada, por ejemplo, 198.162.10.1) puedan salir hacia Internet; pero no pueden aceptar llamadas o servicios del exterior directamente, sino a través de la máquina que tiene la IP real.

Esto significa que algunos servicios no funcionan (por ejemplo, *talk*) y otros deben ser configurados en modo PASV (pasivo) para que funcionen (por ejemplo, FTP). Sin embargo, WWW, telnet o irc funcionan adecuadamente. El *kernel* debe estar configurado con las siguientes opciones: Network firewalls, TCP/IP networking, IP:forwarding/gatewaying, IP: masquerading. Normalmente, la configuración más común es disponer de una máquina con una conexión SLIP o PPP y tener otro dispositivo de red (por ejemplo, una tarjeta Ethernet) con una dirección de red reservada. Como vimos, y de acuerdo a la RFC 1918, se pueden utilizar como IP privadas los siguientes rangos de direcciones (IP/Mask): 10.0.0.0/255.0.0.0, 172.16.0.0/255.240.0.0, 192.168.0.0/255.255.0.0. Los nodos que deben ser ocultados (*masqueraded*) estarán dentro de esta segunda red. Cada una de estas máquinas debería tener la dirección de la máquina que realiza el *masquerade* como *default gateway* o *router*. Sobre dicha máquina podemos configurar:

- Network route para Ethernet considerando que la red tiene un IP = 192.168.1.0/255.255.255.0:

```
route add -net 192.168.1.0 netmask 255.255.255.0 eth0
```

- Default route para el resto de Internet:

```
route add default ppp0
```

- Todos los nodos sobre la red 192.168.1/24 serán *masqueraded*:

```
ipchains -A forward -s 192.168.1.0/24 -j MASQ
```

- Si se utiliza iptables sobre un *kernel* 2.4 o superior:

```
iptables -t nat -A POSTROUTING -o ppp0 -j MASQUERADE
```

Consultar las referencias en la unidad que trata de la de seguridad sobre información de *ipchains* e *iptables*. [Ran05, KD00]

8. NAT con el *kernel* 2.2 o superiores

El IP *network address translation* NAT es el reemplazo que deja obsoleto las prestaciones de GNU/Linux IP Másquerade y que aporta nuevas prestaciones al servicio. Dentro de las mejoras introducidas en la pila de TCP/IP del núcleo 2.2 de GNU/Linux, una es que el NAT forma parte del *kernel*. Para utilizarlo, es necesario que el *kernel* se compile con:

CONFIG_IP_ADVANCED_ROUTER, CONFIG_IP_MULTIPLE_TABLES y CONFIG_IP_ROUTE_NAT.

Y si se necesita control exhaustivo de las reglas NAT (por ejemplo, para activar el cortafuegos *firewalling*) debe estar también

CONFIG_IP_FIREWALL y CONFIG_IP_ROUTE_FWMARK.

Para trabajar con estas nuevas características, es necesario usar el programa ip (se puede obtener en ftp://ftp.inr.ac.ru/ip_routing/). Entonces para trasladar direcciones de datagramas de entrada se puede utilizar:

```
ip route add nat <extaddr>[/<masklen>] via <intaddr>
```

Esto hará que un paquete de entrada destinado a ext-addr (la dirección visible desde fuera de Internet) se transcribe su dirección destino a int-addr (la dirección de su red interna por medio de su *gateway*/*firewall*). El paquete se encamina de acuerdo a la tabla local de *route*. Se pueden trasladar direcciones simples o bloques. Por ejemplo:

ip route add nat 240.0.11.34 via 192.109.0.2
ip route add nat 240.0.11.32/27 via 192.109.0.0

El primero hace que la dirección interna 192.109.0.2 sea accesible como 240.0.11.34. El segundo reubica (*remapping*) el *block* 192.109.0.031 a 240.0.11.3263. En este caso se ha utilizado como ejemplo traslaciones a direcciones de la clase D, E tal como 240.0.*.* con el fin de no utilizar ninguna dirección pública. El usuario deberá reemplazar estas direcciones (240.0.11.34 y 240.0.11.3263) por las correspondientes direcciones públicas a las que desee realizar la traslación. [Ran05]

9. ¿Cómo configurar una conexión DialUP y PPP?

Configurar una conexión dial-up sobre PPP en GNU/Linux es muy simple. PPP (*point to point protocol*) que permite realizar *IP-Links* entre dos ordenadores con un módem (considerar que debe ser un módem soportado por GNU/Linux, ya que no todos, especialmente los internos o los conocidos como Winmodems, se pueden configurar, puesto que muchos de ellos necesitan software adicional para establecer la comunicación). [Vas00, Law07, Sec00].

Como pasos previos se debe disponer de la siguiente información: el init-string del módem (normalmente no es necesario pero si la necesita y no la tiene disponible, se puede utilizar ATZ, que funciona en la mayoría de los módems, o consultar listas especializadas de init-string).

Además, necesitará los datos del ISP: identificación de conexión (*login name*), clave (*password*) y número de teléfono. Direcciones de DNS serían aconsejables, pero es opcional en las versiones actuales de pppd. Verificar además que su módem está correctamente conectado. Con un módem externo se debe ejecutar *echo > /dev/ttyS0* y mirar las luces del módem por si tienen actividad. En caso contrario, intentar con ttyS1 por si el módem está conectado al 2.º puerto serie. Con un módem interno, consultar el manual de hardware soportado para ver si este módem puede ser reconocido por GNU/Linux, y en caso afirmativo, podría tener que reconfigurar el *kernel* para utilizarlo. También puede utilizar *cat /proc/pci* por si se encuentra en el bus PCI. [PPP00]

La forma más fácil de configurar ahora el módem es a través del paquete kppp (debe instalar los paquetes kdenetwork-ppp* y ppp*). Sobre una terminal, ejecútese /usr/bin/kppp. Sobre la ventana, complétense las opciones siguientes:

Accounts ⇒ New Connection
Dial ⇒ Authentication 'PAP/CHAP'
Store Password ⇒ yes
IP ⇒ Dynamic IP Address
Autoconfigure hostname ⇒ No
Gateway ⇒Default *Gateway* ⇒ Assign the Default Route
DNS ⇒Configuration Automatic ⇒Disable existing DNS
Device ⇒ttyS1(com1) o ttyS2 (com2)
Modem ⇒Query Modem para ver los resultados (si no obtiene resultados, cambie el dispositivo ttySx).

Entremos *login* y *password,* y estaremos conectados a Internet (para verificar la conexión podría ejecutar *ping www.google.com* por ejemplo). Aquí se ha utilizado el paquete kppp, pero igualmente podría utilizarse linuxconf o gnome-ppp indistintamente).

Una manera rápida de configurar pppd en Debian consiste en usar el programa pppconfig, que viene con el paquete del mismo nombre. pppconfig configura los archivos como los anteriores después de formular preguntas al usuario a través de una interfaz de menús. Otra opción diferente para usar pppd consiste en ejecutarlo desde wvdial que viene con el paquete wvdial. En vez de hacer que pppd ejecute chat para marcar y negociar la conexión, wvdial realiza el marcado, la negociación inicial y luego inicia pppd para que haga el resto. En la mayoría de los casos dando solamente el número telefónico, el nombre de usuario y la contraseña, wvdial logra establecer la conexión.

Una vez configurado PPP para que funcione por ejemplo con mi_isp, se debe editar /etc/network/interfaces de modo que incluya una sección como la siguiente (los comandos ifup, ifdown utilizan los comandos pon y poff para configurar interfaces PPP):

```
iface ppp0 inet ppp
provider mi_isp
```

con esta sección , ifup ppp0 hace:

```
pon mi_isp
```

Actualmente no es posible usar ifup down para realizar una configuración auxiliar de las interfaces PPP. Como pon desaparece antes que pppd haya terminado de establecer la conexión, ifup ejecuta los *scripts up* antes de que la interfaz PPP esté lista para usar. Hasta que se solucione este fallo, sigue siendo necesario realizar una configuración posterior en /etc/ppp/ip-up o /etc/ppp/ip-up.d/.

Muchos proveedores de servicios de Internet (ISP) de banda ancha utilizan PPP para negociar las conexiones incluso cuando las máquinas de los clientes están conectadas mediante Ethernet y/o redes ATM. Esto se logra mediante PPP sobre Ethernet (PPPoE) que es una técnica para el encapsulamiento del flujo PPP dentro de las tramas Ethernet. Supongamos que el ISP se llama mi_isp. Primero hay que configurar PPP y PPPoE para mi_isp. La manera más fácil de hacerlo consiste en instalar el paquete pppoeconf y ejecutar pppoeconf desde la consola. A continuación, editar /etc/network/interfaces de modo que incluya un fragmento como el siguiente:

```
iface eth0 inet ppp
provider mi_isp
```

A veces surgen problemas con PPPoE relativos a la unidad de transmisión máxima (*maximum transmit unit* o MTU) en líneas DSL (*digital subscriber line*), se puede consultar el DSL-HOWTO para más detalles. También debe tenerse en cuenta si su módem posee un *router*, ya que entonces el *módem/router* maneja por sí mismo la conexión PPPoE y aparece del lado de la LAN como una simple puerta de enlace Ethernet a Internet.

10. Configuración de la red mediante *hotplug*

El paquete *hotplug* permite el soporte de arranque en caliente (se debe tener instalado el paquete del mismo nombre). El hardware de red se puede conectar en caliente ya sea durante el arranque, tras haber insertado la tarjeta en la máquina (una tarjeta PCMCIA, por ejemplo), o después de que una utilidad como discover se haya ejecutado y cargado los módulos necesarios. Cuando el *kernel* detecta nuevo hardware, inicializa el controlador para el hardware y luego ejecuta el programa hotplug para configurarlo. Si más tarde se elimina el hardware, ejecuta nuevamente hotplug con parámetros diferentes. En Debian, cuando se llama a hotplug éste ejecuta los *scripts* de /etc/hotplug/ y /etc/hotplug.d/. El hardware de red recientemente conectado es configurado por el /etc/hotplug/net.agent. Supongamos que la tarjeta de red PCMCIA ha sido conectada, lo que implica que la interfaz eth0 está lista para usar. /etc/hotplug/net.agent hace lo siguiente:

```
ifup eth0=hotplug
```

A menos que haya añadido una interfaz lógica llamada hotplug en /etc/network/interfaces, este comando no hará nada. Para que este comando configure eth0, añadir las siguientes líneas al /etc/network/interfaces:

```
mapping hotplug
script echo
```

Si sólo desea que eth0 se active en caliente y no otras interfaces, utilizar grep en vez de echo como se muestra a continuación:

```
mapping hotplug
script grep
map eth0
```

ifplugd activa o desactiva una interfaz según si el hardware subyacente está o no conectado a la red. El programa puede detectar un cable conectado a una interfaz Ethernet o un punto de acceso asociado a una interfaz Wi-Fi. Cuando ifplugd ve que el estado del enlace ha cambiado, ejecuta un script que por defecto ejecuta ifup o ifdown para la interfaz. ifplugd funciona en combinación con hotplug. Al insertar una tarjeta, lo que significa que la interfaz está lista para usar, /etc/hotplug.d/net/ifplugd.hotplug inicia una instancia de ifplugd para dicha interfaz. Cuando ifplugd detecta que la tarjeta es conectada a una red, ejecuta ifup para esta interfaz.

Para asociar una tarjeta Wi-Fi con un punto de acceso, puede que necesite programarla con una clave de cifrado WEP adecuada. Si está utilizando ifplugd para controlar ifup como se explicó anteriormente, entonces evidentemente no podrá configurar la clave de cifrado usando ifup, ya que éste sólo es llama-

do después de que la tarjeta ha sido asociada. La solución más simple es usar waproamd, que configura la clave de cifrado WEP según los puntos de acceso disponibles que se descubren mediante la búsqueda de la redes WiFi. Para más información consultar man waproamd y la información del paquete.

11. *Virtual private network* (VPN)

Una VPN (*virtual private network*) es una red que utiliza como transporte de datos Internet, pero impide que los datos puedan ser accedidos por miembros externos a ella.

Esto significa tener una red con VPN nodos unidos a través de un túnel por donde viaja el tráfico y donde nadie puede interactuar con él. Es utilizada cuando se tienen usuarios remotos que acceden a una red corporativa para mantener la seguridad y privacidad de los datos. Para configurar una VPN, se pueden utilizar diversos métodos SSH (SSL), CIPE, IPSec, PPTP, que pueden consultarse en la bibliografía (se recomienda consultar VPN PPP-SSH HOWTO, por Scott Bronson, VPN-HOWTO de Matthew D. Wilson). [Bro01, Wil02].

Para llevar a cabo las pruebas de configuración, en este apartado se utilizará la OpenVPN, que es una solución basada en SSL VPN y se puede utilizar para un amplio rango de soluciones, por ejemplo acceso remoto, VPN punto a punto, redes WiFi seguras o redes distribuidas empresariales. OpenVPN implementa OSI layer 2 o 3 utilizando protocolos SSL/TLS y soporta autentificación basada en certificados, tarjetas (smart cards), y otros métodos de certificación. OpenVPN no es un servidor proxy de aplicaciones ni opera a través de una web *browser*.

Para analizarlo utilizaremos una opción de la OpenVPN llamada OpenVPN for Static key configurations, que ofrece una forma simple de configurar una VPN ideal para pruebas o para conexiones punto a punto. Sus ventajas son simplicidad y no es necesario un certificado X509 PKI (*public key infrastructure*) para mantener la VPN. Las desventajas son que sólo permite un cliente y un servidor, al no utilizar llave pública y llave privada puede haber igualdad de llaves con sesiones anteriores, debe existir una llave en modo texto en cada peer, y la llave secreta debe ser intercambiada anteriormente por un canal seguro.

Ejemplo simple

En este ejemplo se configurará un túnel VPN sobre un servidor con IP=10.8.0.1 y un cliente con IP=10.8.0.2. La comunicación será encriptada entre el cliente y el servidor sobre UDP port 1194, que es el puerto por defecto de OpenVPN. Después de instalar el paquete (http://openvpn.net/install.html) se deberá generar la llave estática:

```
openvpn --genkey --secret static.key
```

Después se debe copiar el archivo static.key en el otro peer sobre un canal seguro (por ejemplo, utilizando ssh o scp). El archivo de configuración del servidor openVPN_server por ejemplo:

```
dev tun
ifconfig 10.8.0.1 10.8.0.2
secret static.key
```

El archivo de configuración del cliente por ejemplo openVPN_client

```
remote myremote.mydomain
dev tun
ifconfig 10.8.0.2 10.8.0.1
secret static.key
```

Antes de verificar el funcionamiento de la VPN, debe asegurarse en el *firewall* que el puerto 1194 UDP está abierto sobre el servidor y que la interfaz virtual tun0 usada por OpenVPN no está bloqueada ni sobre el cliente ni sobre el servidor. Tened en cuanta que el 90% de los problemas de conexión encontrados por usuarios nuevos de OpenVPN están relacionados con el *firewall*.

Para verificar la OpenVPN entre dos máquinas, deberéis cambiar las IP por las reales y el dominio por el que tenga, y luego ejecutar del lado servidor

```
openvpn [server config file]
```

El cual dará una salida como:

```
Sun Feb 6 20:46:38 2005 OpenVPN 2.0_rc12 i686-suse-linux [SSL] [LZO]
[EPOLL] built on Feb 5 2005
Sun Feb 6 20:46:38 2005 Diffie-Hellman initialized with 1024 bit key
Sun Feb 6 20:46:38 2005 TLS-Auth MTU parms [ L:1542 D:138 EF:38 EB:0 ET:0
EL:0 ]
Sun Feb 6 20:46:38 2005 TUN/TAP device tun1 opened
Sun Feb 6 20:46:38 2005 /sbin/ifconfig tun1 10.8.0.1 pointopoint 10.8.0.2
mtu 1500
Sun Feb 6 20:46:38 2005 /sbin/route add -net 10.8.0.0 netmask
255.255.255.0 gw 10.8.0.2
Sun Feb 6 20:46:38 2005 Data Channel MTU parms [ L:1542 D:1450 EF:42
EB:23 ET:0 EL:0 AF:3/1 ]
Sun Feb 6 20:46:38 2005 UDPv4 link local (bound): [undef]:1194
Sun Feb 6 20:46:38 2005 UDPv4 link remote: [undef]
Sun Feb 6 20:46:38 2005 MULTI: multi_init called, r=256 v=256
Sun Feb 6 20:46:38 2005 IFCONFIG POOL: base=10.8.0.4 size=62
Sun Feb 6 20:46:38 2005 IFCONFIG POOL LIST
Sun Feb 6 20:46:38 2005 Initialization Sequence Completed
```

Y del lado cliente:

```
openvpn [client config file]
```

Para verificar que funciona, podríais hacer ping 10.8.0.2 desde el server y `ping 10.8.0.1` desde el cliente. Para más información, consultar http://openvpn.net/howto.html.

Para agregar compresión sobre el *link*, debe añadirse la siguiente línea a los dos archivos de configuración:

```
comp-lzo
```

Para proteger la conexión a través de un NAT router/firewall alive y seguir lo cambios de IP a través de un DNS, si uno de los peers cambia, agregar a los dos archivos de configuración:

```
keepalive 10 60
ping-timer-rem
persist-tun
persist-key
```

Para ejecutarse como *daemon* con los privilegios de user/group nobody, agregar a los archivos de configuración:

```
user nobody
group nobody
daemon
```

12. Configuraciones avanzadas y herramientas

Existe un conjunto de paquetes complementarios (o que sustituyen a los convencionales) y herramientas que o bien mejoran la seguridad de la máquina (recomendados en ambientes hostiles), o bien ayudan en la configuración de red (y del sistema en general) en forma más amigable.

Estos paquetes pueden ser de gran ayuda al administrador de red para evitar intrusos o usuarios locales que se exceden de sus atribuciones (generalmente, no por el usuario local, sino a través de una suplantación de identidad) o bien ayudar al usuario novel a configurar adecuadamente los servicios.

En este sentido, es necesario contemplar:

- **Configuración avanzada de TCP/IP**: a través del comando sysctl es posible modificar los parámetros del *kernel* durante su ejecución o en el inicio, para ajustarlos a las necesidades del sistema. Los parámetros susceptibles de modificar son los que se encuentran en el directorio /proc/sys/ y se pueden consultar con sysctl -a. La forma más simple de modificar estos parámetros es a través de archivo de configuración /etc/syscntl.conf. Después de la modificación, se debe volver a arrancar la red:

  /etc/init.d/networking restart

  En este apartado veremos algunas modificaciones para mejorar las prestaciones de la red (mejoras según condiciones) o la seguridad del sistema (consultar las referencias para más detalles) [Mou01]:

  net.ipv4.icmp_echo_ignore_all = 1

- No responde paquetes ICMP como por ejemplo el comando ping que podría significar un ataque DoS (*denial-of-service*).

  net.ipv4.icmp_echo_ignore_broadcasts = 1

- Evita congestiones de red no respondiendo el *broadcast*.

  net.ipv4.conf.all.accept_source_route = 0
  net.ipv4.conf.lo.accept_source_route = 0
  net.ipv4.conf.eth0.accept_source_route = 0
  net.ipv4.conf.default.accept_source_route = 0

- Inhibe los paquetes de IP *source routing*, que podrían representar un problema de seguridad (en todas las interfaces).

  net.ipv4.tcp_syncookies = 1
  net.ipv4.conf.all.accept_redirects = 0

- Permite rechazar un ataque DoS por paquetes SYNC, que consumiría todos los recursos del sistema forzando a hacer un *reboot* de la máquina.

  net.ipv4.conf.lo.accept_redirects = 0
  net.ipv4.conf.eth0.accept_redirects = 0
  net.ipv4.conf.default.accept_redirects = 0

- Útil para evitar ataques con CMP *redirect acceptance* (estos paquetes son utilizados cuando el *routing* no tiene una ruta adecuada) en todas las interfaces.

  net.ipv4.icmp_ignore_bogus_error_responses = 1

- Envía alertas sobre todos los mensajes erróneos en la red.

  net.ipv4.conf.all.rp_filter = 1
  net.ipv4.conf.lo.rp_filter = 1
  net.ipv4.conf.eth0.rp_filter = 1
  net.ipv4.conf.default.rp_filter = 1

- Habilita la protección contra el *IP spoofing* en todas las interfaces.

  net.ipv4.conf.all.log_martians = 1
  net.ipv4.conf.lo.log_martians = 1
  net.ipv4.conf.eth0.log_martians = 1
  net.ipv4.conf.default.log_martians = 1

  Generará *log* sobre todos los *spoofed packets*, *source routed packets* y *redirect packets*.

- Los siguientes parámetros permitirán que el sistema pueda atender mejor y más rápido las conexiones TCP.

  net.ipv4.tcp_fin_timeout = 40, Por defecto, 60.
  net.ipv4.tcp_keepalive_time = 3600, Por defecto, 7.200.
  net.ipv4.tcp_window_scaling = 0
  net.ipv4.tcp_sack = 0
  net.ipv4.tcp_timestamps = 0, Por defecto, todos a 1 (habilitados).

- **Iptables:** las últimas versiones de GNU/Linux (*kernel* 2.4 o superiores) incluyen nuevos mecanismos para construir filtros de paquetes llamado netfilter [Mou01]. Esta nueva funcionalidad es gestionada por una herra-

mienta denominada iptables que presenta mejores características que su predecesor (ipchains). Como se verá en la unidad correspondiente a seguridad, es sumamente fácil construir un *firewall* con esta herramienta para detectar y hacer frente a los ataques más comunes scans, DoS, IP/MAC Spoofing, etc. Su activación pasa primero por verificar que el *kernel* es 2.4 o superior, que el mismo está configurado para dar soporte de ipfilter (lo cual significará que se deberá recompilar el *kernel* para activar la opción *network packet filtering* [*CONFIG_NETFILTER*], y todas las subopciones específicas). Las reglas específicas se deben activar durante el arranque (por ejemplo, a través del /etc/init.d y el enlace adecuado al directorio rc adecuado) y tiene un formato similar (consultar las referencias sobre las capacidades y la sintaxis completa) a:

```
iptables -A Type -i Interface -p protocol -s SourceIP --
source-port Port -d DestinationIP --destination-port Port
-j Action
```

- GnuPG: esta herramienta permite encriptar datos para su posterior envío (por ejemplo correo electrónico) o almacenamiento, y también para generar firmas digitales (cumple con el estándar de la RFC2440) y no utiliza algoritmos patentados, lo cual significa más libertad en el *open source* pero pérdida de compatibilidad con otras herramientas (por ejemplo, PGP 2.0) que utilizan algoritmos como el IDEA y RSA. Para su compilación y/o instalación, seguir las indicaciones de sus autores en http://www.gnupg.org/. En primer lugar, se debe crear una par de claves (pública y privada) ejecutando como *root* el comando `gpg --gen-key` dos veces y contestando las preguntas realizadas por el mismo. Generalmente, estas claves se almacenarán en /*root*. Lo siguiente es exportar (por ejemplo a una página web) la clave pública para que otros usuarios la puedan utilizar para encriptar los correos/información que sólo podrá ver el usuario que ha generado la clave pública. Para ello, habrá que utilizar `gpg --export -ao UID`, lo cual generará un archivo ASCII de la clave pública del usuario UID.

  Para importar una clave pública de otro usuario, se puede usar `gpg --import filename`, y para firmar una clave (significa indicarle al sistema que se está de acuerdo en que la clave firmada es de quien dice ser), se puede utilizar `gpg --sign-key UID`. Para verificar una clave, se puede utilizar `gpg --verify file/data` y para encriptar/desencriptar gpg -sear UID file g, gpg -d file, respectivamente. [Gnu]

- Logcheck: una de las actividades de un administrador de red es verificar diariamente (más de una vez por día) los archivos log para detectar posibles ataques/intrusiones o eventos que puedan dar indicios sobre estas cuestiones. Esta herramienta selecciona (de los archivos log) información condensada de problemas y riesgos potenciales y luego envía esta información al responsable, por ejemplo, a través de un correo. El pa-

quete incluye utilidades para ejecutarse de modo autónomo y recordar la última entrada verificada para las subsiguientes ejecuciones. Para información sobre la configuración/instalación, podéis consultar las referencias. [Log]

- PortSentry y Tripwire: estas herramientas ayudan en las funciones del administrador de red en cuanto a seguridad se refiere. PortSentry permite detectar y responder a acciones de búsqueda de puertos (paso previo a un ataque o a un *spamming*) en tiempo real y tomar diversas decisiones respecto a la acción que se está llevando a cabo. Tripwire es una herramienta que ayudará al administrador notificando sobre posibles modificaciones y cambios en archivos para evitar posibles daños (mayores). Esta herramienta compara las diferencias entre los archivos actuales y una base de datos generada previamente para detectar cambios (inserciones y borrado), lo cual es muy útil para detectar posibles modificaciones de archivos vitales como por ejemplo, en archivos de configuración. Consultar las referencias sobre la instalación/configuración de estas herramientas. [Tri]

- Xinetd: esta herramienta mejora notablemente la eficiencia y prestaciones de inetd y tcp -wrappers. Una de las grandes ventajas de xinetd es que puede hacer frente a ataques de DoA (*denial-of-access*) a través de mecanismos de control para los servicios basados en la identificación de direcciones del cliente, en tiempo de acceso y tiempo de conexión (*logging*). No se debe pensar que Xinetd es el más adecuado para todos los servicios (por ejemplo, FTP y SSH es mejor que se ejecuten solos como *daemons*), ya que muchos de ellos generan una gran sobrecarga al sistema y disponen de mecanismos de acceso seguros que no crean interrupciones en la seguridad del sistema. [Xin]

  La compilación y/o instalación es simple, sólo es necesario configurar dos archivos: /etc/xinetd.conf (el archivo de configuración de Xinetd) y /etc/rc.d/init.d/xinetd (el archivo de inicialización de Xinetd). El primer archivo contiene dos secciones: *defaults*, que es donde se encuentran los parámetros que se aplicarán a todos los servicios y *service*, que serán los servicios que se pondrán en marcha a través de Xinetd.

Un ejemplo típico de la configuración podría ser:

```
# xinetd.conf
# Las opciones de configuración por defecto que se aplican a todos los
# servidores pueden modificarse para cada servicio
defaults
{
 instances = 10
 log_type = FILE /var/log/service.log
 log_on_success = HOST PID
 log_on_failure = HOST RECORD
}
# El nombre del servicio debe encontrarse en /etc/services para obtener
# el puerto correcto
# Si se trata de un servidor/puerto no estándar, usa "port = X"
```

```
service ftp
{
 socket_type = stream
 protocol = tcp
 wait = no
 user = root
 server = /usr/sbin/proftpd
 }
#service telnet
#{
# socket_type = stream
# protocol = tcp
# wait = no
# user = root
# no_access = 0.0.0.0
# only_from = 127.0.0.1
# banner_fail = /etc/telnet_fail
# server = /usr/sbin/in.telnetd
#}
service ssh
{
 socket_type = stream
 protocol = tcp
 wait = no
 user = root
 port = 22
 server = /usr/sbin/sshd
 server_args = -i
}
service http
{
 socket_type = stream
 protocol = tcp
 wait = no
 user = root
 server = /usr/local/apache/bin/httpd
}
#service finger
#{
# socket_type = stream
# protocol = tcp
# wait = no
# user = root
# no_access = 0.0.0.0
# only_from = 127.0.0.1
# banner_fail = /etc/finger_fail
# server = /usr/sbin/in.fingerd
# server_args = -l
#}

# Fin de /etc/xinetd.conf
```

Los servicios comentados (#) no estarán disponibles. En la sección *defaults* se pueden insertar parámetros tales como el número máximo de peticiones simultáneas de un servicio, el tipo de registro (log) que se desea tener, desde qué nodos se recibirán peticiones por defecto, el número máximo de peticiones por IP que se atenderán, o servicios que se ejecutarán como superservidores (imapd o popd), como por ejemplo:

```
default {
instances = 20
log_type = SYSLOG
authpriv log_on_success = HOST
log_on_failure = HOST
only_from = 192.168.0.0/16
per_source = 3
enabled = imaps
}
```

La sección *service*, una por cada servicio como por ejemplo:

```
service imapd {
socket_type = stream
wait = no
user = root
server = /usr/sbin/imapd
only_from = 0.0.0.0/0 #allows every client
no_access = 192.168.0.1
instances = 30
log_on_success += DURATION USERID
log_on_failure += USERID
nice = 2
redirect = 192.168.1.1 993 #Permite redireccionar el tráfico del port 993
hacia el nodo 192.168.1.1
bind = 192.168.10.4
#Permite indicar a cual interfaz está asociado el servicio para evitar problemas de suplan-
tación de servicio.
}
```

El archivo /etc/init.d/xinetd permitirá poner en marcha el servidor (con el enlace adecuado, según el nivel de ejecución seleccionado, por ejemplo, 3, 4 y 5). Es conveniente cambiar los atributos de ambos archivos para garantizar que no son modificados o desactivados con: chmod 700 /etc/init.d/xinetd; chown 0.0 /etc/init.d/xconfig; chmod 400 /etc/xinetd.conf; chattr +i /etc/xinetd.conf.

- Linuxconf: es una herramienta de configuración y administración de un sistema GNU/Linux pero que ha quedado obsoleta si bien se puede encontrar todavía en algunas distribuciones. Más información en http://www.solucorp.qc.ca/linuxconf/.

- Webmin: es otra herramienta (paquetes webmin-core, webmin-dhcp, webmin-inetd, webmin-sshd...) que permite a través de una interfaz web (es necesario tener por ejemplo el servidor Apache instalado), configurar y añadir aspectos relacionados con la red. Si bien se continúa su desarrollo en muchas distribuciones no se incluye por defecto. Más información en http://www.webmin.com/. Para ejecutarla una vez instalada desde un navegador llamar a la URL https://localhost:10000, que solicitará la aceptación del certificado SSL y el usuario (inicialmente *root*) y su clave (*passwd*).

- System-config-*: en Fedora existe una variedad de herramientas gráficas que se llaman system-config-"alguna-cosa" y donde “alguna-cosa” es para lo que están diseñadas. En general, si se está en un entorno gráfico se puede llegar a cada una de ellas por medio de un menú, sin embargo, cada una de estas herramientas implica un menú a recordar. Una herramienta que centraliza todas las system config es system-config-control en una sola entrada de menú y en una única interfaz gráfica desde la cual se puede seleccionar de acuerdo a una organización de iconos. Para ello, es necesario Applications -> Add/Remove Software y éste se arranca como *root* en el gestor gráfico de software Pirut (se debe tener habilitado el repositorio Fedora Extras). En la interfaz de Pirut, se usa por ejemplo la búsqueda de paquetes

disponibles con el patrón system-config-* haced vuestra selección de system-config-control* y haced un clic en Apply. Entre otras opciones allí se podrán configurar casi todos los aspectos de red y servicios.

- Networkmanager: es una herramienta que permite manejar fácilmente redes inalámbricas y por cable en forma simple y sin grandes complicaciones, pero no es indicado para servidores (sólo para máquinas de escritorio). Su instalación es muy fácil apt-get install network-manager-xx, donde xx es gnome o kde según el escritorio instalado. Para configurarlo, se deben comentar todas las entradas en (Debian) /etc/network/interfaces excepto la interfaz de loopback interface, por ejemplo dejando sólo:

```
auto lo
iface lo inet loopback
```

  Este paso no es obligatorio, pero acelera el descubrimiento de las redes/interfaces. Sobre Debian se debe también agregar un paso extra y es que el usuario debe integrarse dentro del grupo netdev por una cuestión de permisos. Para hacerlo, habrá que ejecutar (como *root* o si no con el comando sudo por delante) `adduser usuario_actual netdev` y hacer un reboot (o también reiniciar la red con `/etc/init.d/networking restart` y hacer un logout-login –salir y entrar– para que el usuario actual se quede incluido en el grupo netdev).

- Otras herramientas: (algunas de ellas están recogidas en la unidad que trata acerca de la seguridad) Nmap (explorar y auditar con fines de seguridad una red), Nessus (evaluar la seguridad de una red de forma remota), Wireshark http://www.wireshark.org/download.html (ex-Ethereal) (analizador de protocolos de red), Snort (sistema de detección de intrusos, IDS), Netcat (utilidad simple y potente para depurar y explorar una red), TCPDump (monitorización de redes y adquisición de información), Hping2 (genera y envía paquetes de ICMP/UDP/TCP para analizar el funcionamiento de una red).

Actividades

1. Definir los siguientes escenarios de red:
a) Máquina aislada.
b) Pequeña red local (4 máquinas, 1 *gateway*).
c) 2 redes locales segmentadas (2 conjuntos de 2 máquinas y un *router* cada una y un *gateway* general).
d) 2 redes locales interconectadas (dos conjuntos de 2 máquinas + *gateway* cada una).
e) 2 máquinas conectadas a través de una red privada virtual. Indicar la ventajas/desventajas de cada configuración, para qué tipo de infraestructura son adecuadas y qué parámetros relevantes se necesitan.

2. Configurar la red de la opción a, b y d del punto.

Anexo. Controlando los servicios vinculados a red en FC6

Un aspecto importante de todos los servicios es cómo se ponen en marcha. FC6 incluye una serie de utilidades para gestionar los servicios –daemons– (incluidos los de red). Como ya se ha visto en el capítulo de administración local, el *runlevel* es el modo de operación que especifica qué *daemons* se ejecutarán. En FC podemos encontrar: runlevel 1 (monousuario), runlevel 2 (multiusuario), runlevel 3 (multiusuario con red), runlevel 5 (X11 más /runlevel 3). Típicamente se ejecuta el nivel 5 o 3 si no se necesitan interfaces gráficas. Para determinar qué nivel se está ejecutando, se puede utilizar /sbin/runlevel, y para saber qué nivel es el que se arranca por defecto `cat /etc/inittab | grep :initdefault:` que nos dará información como id:5:initdefault: (también se puede editar el /etc/inittab para cambiar el valor por defecto.

Para visualizar los servicios que se están ejecutando, podemos utilizar `/sbin/chkconfig –list` y para gestionarlos podemos utilizar system-config-services en modo gráfico o ntsysv en la línea de comandos. Para habilitar servicios individuales, podemos utilizar chkconfig, por ejemplo el siguiente comando habilita el servicio crond para los niveles 3 y 5: `/sbin/chkconfig --level 35 crond on`. Independientemente de cómo se hayan puesto en marcha los servicios se puede utilizar /sbin/service –status-all o individualmente /sbin/service crond status para saber cómo está cada servicio. Y también gestionarlo (start, stop, status, reload, restart), por ejemplo service crond stop para pararlo o service crond restart para reiniciarlo.

Es importante no deshabilitar los siguientes servicios (a no ser que se sepa lo que se está haciendo): acpid, haldaemon, messagebus, klogd, network, syslogd. Los servicios más importantes vinculados a la red (si bien no todos y no es una lista exhaustiva, se recogen la mayoría de ellos) son:

NetworkManager, NetworkManagerDispatcher: es un *daemon* que permite cambiar entre redes fácilmente (Wifi y Ethernet básicamente). Si sólo tiene una red no es necesario que se ejecute.

avahi-daemon, avahi-dnsconfd: es una implementación de zeroconf y es útil para detectar dispositivos y servicios sobre redes locales sin DNS (es lo mismo que mDNS).

bluetooth, hcid, hidd, sdpd, dund, pand: Bluetooth red inalámbrica es para dispositivos portátiles (no es wifi,802.11). Por ejemplo, teclados, *mouse*, teléfonos, altavoces/auriculares, etc.

capi, isdn: red basada en hardware ISDN (RSDI en castellano).

Iptables: es el servicio de *firewall* estándar de Linux. Es totalmente necesario por seguridad si se tiene conexión a red (cable, DSL, T1).

Ip6tables: idem anterior pero para el protocolo y redes basadas en Ipv6.

netplugd: Netplugd puede monitorizar la red y ejecutar comando cuando su estado cambie.

netfs: se utiliza para montar automáticamente sistemas de archivos a través de la red (NFS, Samba, etc.) durante el arranque.

nfs, nfslock: son los *daemon* estándar para compartir sistemas de archivos a través de la red en sistemas operativos estilo Unix/Linux/BSD.

ntpd: servidor de hora y fecha a través de la red.

portmap: es un servicio complementario para NFS (*file sharing*) y/o NIS (*authentication*).

rpcgssd, rpcidmapd, rpcsvcgssd: se utiliza para NFS v4 (nueva versión de NFS).

sendmail: este servicio permite gestionar los mails (MTA) o dar soporte a servicios como IMAP o POP3.

smb: este *daemon* permite compartir ficheros sobre sistemas Windows.

sshd: SSH permite a otros usuarios conectarse interactivamente de forma segura a la máquina local.

yum-updatesd: servicio de actualizaciones por red de FC.

xinetd: servicio alternativo de inetd que presenta un conjunto de características y mejoras, como por ejemplo lanzar múltiples servicios por el mismo puerto (este servicio puede no estar instalado por defecto).